Multi-theorem Designated-Verifier NIZK for QMA

Abstract. We present a designated-verifier non-interactive zero-knowledge argument system for QMA with multi-theorem security under the Learning with Errors Assumption. All previous such protocols for QMA are only single-theorem secure. We also relax the setup assumption required in previous works. We prove security in the malicious designated-verifier (MDV-NIZK) model (Quach, Rothblum, and Wichs, EUROCRYPT 2019), where the setup consists of a mutually trusted random string and an untrusted verifier public key.

Our main technical contribution is a general compiler that given a NIZK for NP and a quantum sigma protocol for QMA generates an MDV-NIZK protocol for QMA.

1   Introduction

Zero-knowledge protocols allow to prove statements without revealing anything but the mere fact that they are true. Since their introduction by Goldwasser, Micali, and Rackoff [GMR89] they have had a profound impact on modern cryptography and theoretical computer science at large. While standard zero-knowledge protocols are interactive, Blum, Feldman, and Micali [BFM19] introduced the concept of a non-interactive zero-knowledge (NIZK) protocol, which consists of a single message sent by the prover to the verifier. NIZK protocols cannot exist in the plain model (i.e. a language with such a NIZK protocol can be decided by an efficient algorithm) but can be realized with a pre-computed setup. The point of the setup is that it can be computed instance-independently and usually, the setup is executed by a trusted third party that generates and publishes a string of bits and sometimes trapdoors are handed to the prover or verifier (or both).

Although existing zero-knowledge protocols for NP cover an array of diverse tasks and in particular, under standard computational assumptions it is known how to construct NIZK protocols for NP [CCH+19, PS19, BKM20], far less is known about the class QMA, the quantum generalization of NP. This knowledge gap between NP and QMA, which is present in both interactive and non-interactive zero-knowledge protocols, stems from the fact that many of the techniques that work for constructing protocols for NP, implicitly rely on the assumption that information in the system is classical. Accordingly, these techniques fail when this assumption no longer holds.

The first gap between classical and quantum NIZK protocols is that of setup requirements, that is, how much trust and resources the setup needs. The standard setup in NIZK is called the common reference string (CRS) model, where the trusted party samples a classical string from some specified distribution and publishes it. If the reference string is simply uniformly random then the setup is in the common *random* string model, which is considered to require minimal trust in the NIZK setting. While NIZK arguments for NP are known to exist in the common random string model under LWE [CCH+19, PS19], in current QMA constructions the setup is comprised at least of a common reference string sampled by the trusted party, and an additional public and secret verification keys (pvk, svk) where pvk is published along with the CRS and svk is kept by the verifier, such that either:

- pvk is a quantum state that needs to stay coherent while waiting for the proof by the prover, or
- The pair (pvk, svk) can be sampled only by the trusted party and not the verifier.

Aside from the above, a more elementary missing part in current NIZK protocols for QMA is multi-theorem security, which provides the main efficiency advantage to a NIZK protocol over an interactive protocol. Multi-theorem security considers the *reusability of the setup*, that is, once the setup is computed, any prover can send a proof by a single message repeatedly for many different statements and there is no need to re-compute the setup for every new proof sent, and in relation to the above QMA setups: once the CRS and public verification key are published, they are reusable.

Given the gap of knowledge in NIZK techniques between NP and QMA, improving the power of NIZKs for QMA and specifically constructing a reusable non-interactive

zero-knowledge protocol for QMA seem as a natural cryptographic goal which we explore in this work.

1.1 Results

Under the Learning with Errors (LWE) assumption [Reg09] we resolve the above question. Specifically, we construct a multi-theorem-secure NIZK argument for QMA in the malicious designated-verifier model, which is the following:

1. The trusted party samples only a common random string crs.
2. Given crs, any verifier can sample a pair of classical public and secret verification keys (pvk, svk), in particular it is possible that the published pvk is maliciously-generated.

Given crs and pvk, any prover can repeatedly give a non-interactive zero-knowledge proof by a single quantum message $|\pi\rangle$. The MDV-NIZK model is introduced by Quach, Rothblum, and Wichs in [QRW19] and has the same minimal trust requirements as the common random string model (but is privately verifiable).

Theorem 1 (informal). *Assuming that LWE is hard for polynomial-time quantum algorithms, there exists a reusable, non-interactive computational zero-knowledge argument system for QMA in the malicious designated-verifier model.*

Main Technical Contribution: General Sigma Protocol MDV-NIZK Compilation. Technically, we deviate from previous NIZK constructions for QMA and aim for a simple and classical technique which is post-quantum (i.e. preserves security also for quantum protocols). Specifically, our main contribution is showing how given a NIZK for NP it is possible to compile a quantum sigma protocol into a reusable MDV-NIZK protocol. Further details are given in the technical overview below.

1.2 Technical Overview

We next describe our construction of a multi-theorem-secure MDV-NIZK protocol for QMA. For a discussion about the possibility of constructing a NIZK protocol for QMA in the CRS model see subsection 1.3, and for an overview of NIZK models and previous work on NIZK for QMA see subsection 1.3.

As we are aiming for a classical (quantum-secure) technique we currently restrict our attention to a purely-classical question: Given any sigma protocol $(\Sigma.\mathsf{P}, \Sigma.\mathsf{V})$, generically compile it into a multi-theorem-secure MDV-NIZK while assuming minimal properties of the protocol[1]. We will start with considering classical sigma protocols and later see what changes should take place in order for the technique to work for quantum protocols.

From a Sigma Protocol to a Single-Theorem-Secure MDV-NIZK. A sigma protocol is a 3-message public-coin proof system (with some mild zero knowledge properties),

---

[1]In particular, we do not assume that the message α is classical.

where the 3 messages are denoted by α, β and γ (i.e. β is a random string and is called "the challenge string"). Our first step is to construct a MDV-NIZK protocol with only single-theorem security out of a sigma protocol and is very simple.

In a sigma protocol, since the verifier's message β is a random string it is independent of any other information, additionally, our second need from it is that it stays hidden (until after the prover sends its first message α). The verifier can compute its public verification key, which is computed instance-independently, as a function of β: The public verification key pvk is an FHE-encrypted random challenge β and the secret verification key svk is the FHE decryption key and the challenge string,

$$\mathsf{pvk} = \mathsf{FHE.Enc}_{\mathsf{fhek}}(\beta), \quad \mathsf{svk} = (\beta, \mathsf{fhek}) \ .$$

Given the public verification key pvk, the 1-message proof procedure for $x \in \mathcal{L}$ goes as follows:

- P computes the first sigma protocol message $\alpha \leftarrow \Sigma.\mathsf{P}(x, w)$, where $w \in \mathcal{R}_{\mathcal{L}}(x)$.
- P computes γ the last protocol message under the encryption, that is, P performs the homomorphic evaluation $\hat{\mathsf{ct}}_{\mathsf{P}} \leftarrow \mathsf{FHE.Eval}(\Sigma.\mathsf{P}_3, \mathsf{FHE.Enc}_{\mathsf{fhek}}(\beta))$.
- As the proof, P sends α out in the open and γ under the encryption, that is, the proof is $\pi = (\alpha, \hat{\mathsf{ct}}_{\mathsf{P}})$.

In order for the proof to stay zero-knowledge, the homomorphic evaluation needs to be circuit-private. The verification algorithm is straightforward: Given svk, an instance x and a proof $\pi = (\alpha, \hat{\mathsf{ct}}_{\mathsf{P}})$, the verifier decrypts $\hat{\mathsf{ct}}_{\mathsf{P}}$ to get γ, and accepts iff the sigma protocol verifier accepts $\Sigma.\mathsf{V}(x, \alpha, \beta, \gamma) = 1$.

Is the Above Protocol Multi-Theorem-Secure? While it is intuitively clear that the described construction is secure for a single use of the setup (that is, the above should, with some modifications, yield a single-theorem-secure MDV-NIZK) it is provably not multi-theorem-secure. Sigma protocols are usually parallel repetitions of 3-message zero-knowledge protocols, for example, consider the sigma protocol which is the parallel repetition of the zero-knowledge protocol for Graph Hamiltonicity [Blu86], which is as follows: Given a Hamiltonian cycle C in a graph $G = (V, E)$, the prover samples a random permutation $\varphi : V \rightarrow V$ of the vertices and commits to the permuted graph $\varphi(G)$[2]. The verifier then sends a random bit b, and the prover answers accordingly:

- If $b = 0$ it is considered as a validity check, and the prover opens all commitments and sends φ. The verifier accepts if indeed the committed graph is $\varphi(G)$.
- If $b = 1$ it is considered as the cycle check, and the prover opens commitments only for the subgraph $\varphi(C)$. The verifier accepts if the opening shows a Hamiltonian cycle.

If the sigma protocol used in the above MDV-NIZK construction is the parallel repetition of the zero-knowledge protocol for Hamiltonicity[3], then there is a polynomial-time

---

[2]That is, the prover commits to all of the cells in the adjacency matrix that represents the graph $\varphi(G)$.

[3]We take the Hamiltonicity protocol only as a concrete easy example and in fact any other sigma protocol can take the role of this protocol in our context of attacking the soundness.

malicious prover P^* that given multiple access to the verifier's verdict function $V(svk, \cdot)$ using the same public/secret verification key pair, can decode the encrypted challenge string β (which is polynomially-many random bits, each bit is for the i-th parallel repetition of the zero-knowledge protocol) and consequently break the soundness.

P^* takes a Hamiltonian graph G and a Hamiltonian cycle C in it, and will decode the entire $\beta = (b_1, b_2, \cdots, b_k)$ bit-by-bit: To decode b_i, P^* will honestly execute the zero-knowledge protocol prover's algorithm for all indices but index i (that is, for all $j \neq i$, it will honestly compute $\mathsf{Com}(\varphi_j(G))$ and under the encryption, the opening of either the entire graph and the permutation of just the cycle $\varphi_j(C)$), for which it is going to operate as follows. P^* will guess that $b_i = 0$ and send a commitment to a permutation of the graph out in the open and under the encryption act as if $b_i = 0$ regardless of the actual value of b_i. By the verifier's acceptance or rejection it will know whether the bit was 0 or 1. After decoding β the prover can now use this information to "prove" that any graph G is Hamiltonian.

From Single-Theorem to Multi-Theorem Security. In the above attack the prover heavily relied on a specific operation: It uses a yes-instance (in the above case, a Hamiltonian graph G), in order to decode the random challenge β and then goes on to use the knowledge of β to give a false proof for a no-instance (again, in the above, a non Hamiltonian graph G^*).

Crucially, P^* does not know how to decode β when the graph is not Hamiltonian. More specifically, in the above we decode β bit-by-bit rather than all at once, and this ability comes from the fact that G is Hamiltonian and the zero-knowledge protocol is complete, thus P^* can be sure that if it honestly executes the zero-knowledge protocol for all indices but i, the only index that can make the proof get rejected is i. In this isolation, checking whether the challenge bit b_i is 0 or 1 becomes easy. However, if the graph is not Hamiltonian then the prover cannot know which index made the proof get rejected because all k indices are prone to rejection. Formally, by the soundness of the sigma protocol, we know that the answer from the verdict function of the verifier in this case will always be a rejection for any polynomial (or even sub-exponential) number of queries, with overwhelming probability. This means in particular that the prover cannot decode anything through the oracle access to the verdict function.

Our fix to the first protocol is based on the above observation: If we could make the random challenge β change with the instance at hand it seems that the decoding attack is neutralized, because even if the prover decodes β_G the challenge for a Hamiltonian graph G, it doesn't have information about β_{G^*} the challenge of some non Hamiltonian G^*. Since the instance x *is in particular a classical string* we can make the challenge change with the instance: The public verification key will not be an encrypted challenge β but instead will be a secret key prfk of a pseudorandom function PRF. The prover will compute α out in the open as before but the homomorphic evaluation changes: under the encryption, P will compute the challenge string as the PRF's output on the instance $\beta_x = \mathsf{PRF}.\mathsf{F}_{\mathsf{prfk}}(x)$, and then compute γ for the challenge β_x.

Extraction by Non-interactive Zero Knowledge for NP. Up to this point we only came close to constructing a provably-secure MDV-NIZK. Indeed, we didn't even use any NIZK tools yet for NP, and in order to prove the security of our construction we need knowledge extraction from both the prover and verifier.

To prove soundness, our thought process is roughly the following: We know that the prover computes γ obliviously under the FHE, more precisely, it homomorphically evaluates the circuit $C_{x,r}$ that computes $\beta_x = \mathsf{PRF.F}_{\mathsf{prfk}}(x)$ and then given β_x computes γ. The part of the circuit $C_{x,r}$ that computes γ from β_x is the "non-trivial" part of the circuit and is determined by a secret string r (which is the information that the honest sigma protocol prover uses in order to compute γ, this information is the randomness of the prover and possibly the witness). If we could extract r from a prover (e.g. by the prover giving a proof of knowledge on the non-trivial part of the circuit $C_{x,r}$) that successfully cheats in the NIZK protocol then we could get a successfully cheating prover for the sigma protocol and thus prove security. To see this, note that by the hiding of the FHE and by the pseudorandomness of the PRF, even if as the public verification key we send an encryption of 0 instead of an encryption of the PRF secret key, the string r still needs to yield a circuit $C_{x,r}$ that does well in generating a satisfying γ for a now-truly-random challenge β.

On the zero knowledge side we also need extraction; we recall a basic property of a sigma protocol: if the sigma protocol simulator knows the challenge string β before sending the first message α then it can simulate a view that is indistinguishable from the real interaction with the honest prover. This means that the information we want to extract from the malicious verifier is the secret PRF key prfk that in particular holds the information for obtaining β_x.

We solve both extraction tasks by a combination of a two-sided NP NIZK and a public-key encryption scheme with pseudorandom public keys. Given the existence of a PKE scheme $(\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with pseudorandom public keys of length ℓ we take the common random string of our protocol to be (1) the common random string of an NP NIZK $(\mathsf{NIZK.Setup}, \mathsf{NIZK.P}, \mathsf{NIZK.V})$ protocol which we denote with crs, concatenated with (2) a random string of length ℓ which we denote with ek (for extraction key).

We will let each of the parties encrypt, using $\mathsf{PKE.Enc}_{\mathsf{ek}}(\cdot)$, the secrets that we want to extract and then use the NIZK to prove consistency between the content of the PKE encryption and the protocol computations. More precisely, as part of its 1-message proof, the prover will give a proof π_P that the string r encrypted using the PKE yields the (canonical) circuit $C_{x,r}$ that it used for the (circuit-private) homomorphic evaluation that generated γ, and the verifier, as part of its public verification key, will give a proof π_V that the PRF key prfk that is encrypted using the PKE is the same key encrypted with the FHE. Note that the information that the parties encrypt using a random string instead of a real PKE key stays secure due to the fact that a real key is indistinguishable from a random string, and thus an adversary that manages to break the PKE when it uses a random string as the public key can break the pseudorandomness property of the public keys.

When wanting to extract information (either in the soundness reduction or in the zero-knowledge simulation), we will sample ek using the PKE key-generation algorithm $(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}$, and since the public keys are pseudorandom the change in key distribution won't be felt by either of the parties. At that point the parties encrypt their secrets and prove they do so using the NIZK, and the extractor can just use the PKE decryption $\mathsf{PKE.Dec}_{\mathsf{sk}}(\cdot)$ to obtain the secrets.

Compiling Quantum Protocols. Our technique so far is entirely classical and compiles classical sigma protocols. We now ask whether it works to compile quantum sigma protocols. This can be answered in turn by answering the following question: what properties of the sigma protocol *exactly* did we use in order for the MDV-NIZK protocol to work?

It can be verified that even if we don't assume nothing on the sigma protocol that we compile, every action in the MDV-NIZK protocol except the homomorphic evaluation of the circuit $C_{x,r}$ can stay exactly the same. Regarding the homomorphic evaluation, the issue that we have is the following: In order to still be able to extract the information r of the circuit $C_{x,r}$ from the prover, the computation that takes β_x and outputs γ needs to be a classical circuit. This is not necessarily the case in a quantum protocol. For example, in the quantum zero-knowledge protocol for QMA of [BJSW16] (which is also the basis for the quantum NIZK protocol of [CVZ19]), in order to generate γ given α, β, first a quantum Clifford operation that is chosen with respect to β needs to be executed on α, followed by a measurement. Then, the prover proves in ZK that the classical string obtained by the measurement satisfies some properties[4]. With this goal in mind, we identify a different quantum protocol that in fact does satisfy the property that γ can be computed by an entirely classical circuit.

We consider the Consistency of Local Density Matrices (CLDM) problem [Liu06], which is a QMA problem with some special properties. In [BG19] Broadbent and Grilo show that CLDM is QMA-complete and how to construct a very simple quantum zero-knowledge protocol for it. The [BG19] zero-knowledge protocol for CLDM is as follows: Given a quantum witness $|w\rangle$, the protocol starts with the prover sending a quantum one-time pad encryption of $|w\rangle$ as the message α. More precisely, for a length-l witness it samples classical random pads $a, b \leftarrow \{0, 1\}^l$, applies

$$\bigotimes_{i \in [l]} \left(X^{a_i} \cdot Z^{b_i} \right) \cdot |w\rangle \ ,$$

and then sends as α the transformed quantum state and classical commitments to the QOTP keys a, b. For a random challenge β, the prover response γ is an opening to part of the state. We find the CLDM problem and specifically the zero-knowledge protocol for it especially attractive for our purposes as γ *is only a function of the randomness of the prover and the challenge* β, which in particular means that the circuit $C_{x,r}$ can stay classical in our setting.

Finally, by using the sigma protocol yielded by the parallel repetition of the zero-knowledge protocol from [BG19] we obtain a clean and simple non-interactive computational zero-knowledge argument system for the class QMA in the malicious designated-verifier model:

1. **Common Random String:** $(\mathsf{crs}, \mathsf{ek})$.
2. **Public and Secret Verification Keys:** $\mathsf{prfk} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, $\mathsf{fhek} \leftarrow \mathsf{FHE.Gen}(1^\lambda)$,

$$\mathsf{pvk} = \left(\mathsf{FHE.Enc}_{\mathsf{fhek}}(\mathsf{prfk}), \ \mathsf{PKE.Enc}_{\mathsf{ek}}(\mathsf{prfk}), \ \pi_\mathsf{V} \right), \ \ \mathsf{svk} = \left(\mathsf{prfk}, \ \mathsf{fhek} \right) \ .$$

---

[4]in that protocol it is also needed that the verifier itself makes the Clifford operation and measurement, which makes the protocol more challenging to use for a NIZK protocol.

For any prover that wishes to give a proof for an instance $x \in \mathcal{L}_{yes}$, it executes the following:

- **Proof:** If π_V is valid, P computes $\alpha \leftarrow \Xi.\mathsf{P}(|w\rangle; r)$ and sends

$$|\pi\rangle = \big(\alpha, \; \mathsf{FHE.Eval}(C_{x,r}, \mathsf{FHE.Enc}(\mathsf{prfk})), \; \mathsf{PKE.Enc_{ek}}(r), \; \pi_P\big) \; .$$

1.3   Related Work

In this section we discuss the main challenges in the construction of non-interactive zero-knowledge protocols for QMA (specifically in the CRS model) and the previous works on QMA NIZKs.

Can we Build a NIZK protocol for QMA in the CRS model? In short, the answer to the above question is that we don't know, and this section does not aim to answer it. This section is intended to give some evidence to why constructing a NIZK for QMA in the CRS model seem to require a different set of techniques from what we currently have for NP. In what follows we will start with briefly recalling how NIZKs for NP are constructed and then understand why current approaches fail in the setting of quantum proofs.

NP, Fiat-Shamir and Correlation Intractability. In order to construct a non-interactive zero-knowledge protocol for NP under standard assumptions, the construction starts with a sigma protocol $(\Sigma.\mathsf{P}, \Sigma.\mathsf{V})$. To make the protocol non-interactive, the Fiat-Shamir transform is applied: By assuming public oracle access to a random function F, the prover applies it to α and treat its (random-string) output $F(\alpha)$ as the challenge string β. It then computes γ and sends all of this information to the verifier, who makes sure that β was rightfully generated $\beta = F(\alpha)$, and that the sigma protocol verifier $\Sigma.\mathsf{V}(\alpha, \beta, \gamma)$ accepts. Since we don't know how to construct a cryptographic primitive that acts as a publicly-computable random function, the above protocol is secure only in the random oracle model, that is, only if we directly assume public access to such random function F.

In order to prove the security of the NIZK protocol in the standard model (with access to a common reference string rather than a random oracle), the final part of the construction involves swapping the random function F with a new, special hash function H - this general technique of swapping F with a special hash function H is usually called the Correlation Intractability (CI) paradigm [CGH04]. The properties of the hash function H or the meaning of correlation intractability are less relevant to this overview, but it is suffices to say that under the LWE assumption it is known how to construct a hash function H that can be swapped with F in the FS transform and where the protocol can be proven secure [CCH$^+$19, PS19].

Can we use Known Classical NIZK Techniques for Quantum Protocols? There are two known routes for getting a quantum-secure NIZK for NP in the CRS model, the first is through the FS transform and CI (which also uses only standard assumptions, described above) and the second is through the hidden bits model and indistinguishability obfuscation. It is natural to ask whether we can use these techniques for QMA (the

question of whether the FS transform can be used for quantum protocols was asked as one of the open questions in section 1.4 of [BG19]).

We first review the ability to use the FS transform (and in particular correlation intractability) for QMA and explain why there is an issue with the no-cloning theorem. In the quantum setting, sigma protocols $(\Xi.\mathsf{P}, \Xi.\mathsf{V})$ [BG19, BJSW16] are quite the same but with the main difference that the first message α is quantum (and of course, the prover takes as input a quantum witness $|w\rangle$ rather than classical). Recall that when we use the FS transform on a sigma protocol in order to generate a NIZK, for the protocol to be complete, when the parties act honestly then the verifier needs to verify that the random function F yields the challenge, that is $F(\alpha) = \beta$. This means that now F needs to be a quantum transformation such that for $x \in \mathcal{L}_{yes}$ and an honestly generated $\alpha \leftarrow \Xi.\mathsf{P}(|w\rangle)$, $F(\alpha)$ is always the same classical string (with overwhelming probability). Also, for the protocol to be sound we need that the *entire* output of F will be the challenge β and it cannot be the case for example that the output $F(\alpha)$ will contain one register with the classical string β and another register with some quantum state $|\psi\rangle$. Now, denote by s the classical string s.t. $F(\alpha) = s$, and we have a generating circuit for the quantum witness: $|w\rangle = \Xi.\mathsf{P}^\dagger(\cdot) \cdot F^\dagger \cdot |s\rangle$, where the inverse versions of F and $\Xi.\mathsf{P}$ are purified. This seems to violate the no-cloning theorem in the following manner: the prover gets a copy of the witness and can generate a generating circuit for the witness state, this circuit can be used to generate arbitrarily many copies of the state. Finally, because we can always consider a trivial language with a dummy witness, and take the quantum witness to be some unclonable state (for example, a pseudorandom quantum state) we get a contradiction to the no-cloning theorem.

Even if we aim to construct a NIZK using the FS transform for QCMA, the subclass of QMA where the verification algorithm is still quantum but the witness is classical, the problem is not seemed to be solved. The reason, is that we don't know how to construct sigma protocols for QCMA where the first message α is classical, and the same contradiction to the no-cloning theorem holds.

The second known route of obtaining a quantum-secure NIZK protocol for NP in the CRS model is through the hidden bits model [FLS99] which is implementable by sub-exponentially-secure indistinguishability obfuscation [BP15]. In the hidden bits model, intuitively (and roughly), the trusted party samples as the common reference string a commitment to a string sampled from some distribution (where by using a trapdoor permutation, the prover can open the commitments efficiently), and the prover proves that the instance at hand $x \in \mathcal{L}_{yes}$ satisfies some property related to the string underlying the commitments. Even if we are willing to assume the very strong cryptographic assumptions which are needed for the realization of this protocol (i.e. sub-exponentially-secure post-quantum indistinguishability obfuscation), it is currently unknown how to use the hidden bits model to instantiate non-interactive zero-knowledge quantum protocols.

Relaxations of the CRS Model and Previous Work The constructions of NIZKs for NP discussed in subsection 1.3 are implicitly in the CRS model, where the setup consists of a string that is sampled and published by the trusted party, in particular, nor the prover or verifier hold any trapdoors over the setup. Sometimes when it is unknown how to

build a NIZK in the CRS model (or unknown how to minimize the assumptions for building one) we turn to relaxations of the CRS model. For example, in the designated-verifier model (DV-NIZK) [PV+06] the trusted party samples, along with the CRS, a pair of public and secret verification keys (pvk, svk), publishes pvk along with the CRS and hands svk only to the verifier. Another example is the designated prover model (DP-NIZK) [KW19], which is analogous to the DV-NIZK model, only that the prover is the one who gets a secret, now-*proof* key.

It is a well known fact in the design of NIZKs that when the verifier holds a secret verification key (e.g. in the DV-NIZK model) then multi-theorem zero knowledge can be achieved generically by the compiler of [FLS99], but multi-theorem soundness becomes non-trivial. For example, it is possible (and is sometimes provably the case) that the prover can decode the verifier's secret key by having access multiple times to the verifier's verdict function, consequently breaking the soundness of the protocol. Indeed, one example is that until the works of [QRW19, LQR+19], based on [PV+06] it was only known how to get single-theorem-secure DV-NIZK for NP, and another example is that this is the current situation with QMA constructions of NIZK protocols.

The QMA NIZK protocol of Broadbent and Grilo [BG19] is in the secret parameters model (i.e. the protocol is both designated-prover and designated-verifier and both parties get secret keys from the trusted party) but is a proof system and has *statistical* soundness rather than the computational soundness we achieve. The protocol of Coladangelo, Vidick and Zhang [CVZ19] is in a model that is somewhat between the common reference string model and the DV-NIZK model, where the trusted party samples a common reference string and the verifier itself samples a pair (pvk, svk) where pvk is a quantum state. Morimae [Mor20] shows a classical-designated-verifier NIZK proof system for QMA with a quantum trusted setup. Outside of the standard model, an additional construction by Alagic, Childs, Grilo and Hung [ACGH19] yields a QMA NIZK argument in the quantum random oracle model (with additional setup in the secret parameters model) which is classical-verifier. All of the abovementioned protocols are not reusable.

There are two main issues with letting the trusted party sample secret keys for any of the parties: First, the trust requirements of the setup now increase as the party receiving the secret key should assume that the trusted party handles its secret information securely. The second issue is that of centralization of computational resources: for example, in the DV-NIZK model, the trusted party is now responsible for sampling a fresh pair (pvk, svk) for every new verifier that wishes to use the protocol, which is very different from the CRS setting where it samples a string and from that point on can terminate.

The *malicious* designated-verifier (MDV-NIZK) model [QRW19, LQR+19] seeks to solve the above two problems, which is also the model of our protocol. In the MDV-NIZK model the trusted party only samples a common random string, and then, any verifier wishing to use the protocol can sample *by itself* a pair of classical keys (pvk, svk) and publish pvk. The protocol then stays secure even if the public key pvk is maliciously-generated.

1.4 Subsequent Work

Subsequently to this work, several related constructions for NIZK protocols for QMA are shown. Morimae and Yamakawa construct a classically verifiable dual-mode NIZK for QMA, with quantum preprocessing [MY21]. The result is essentially in the same model of the protocol of Coladangelo, Vidick and Zhang [CVZ19], that is, the CRS is published, and then the verifier sends an instance-independent quantum message to the prover, keeping a classical trapdoor. Given the instance, witness and the verifier's quantum message, the prover can perform an efficient quantum procedure and send a classical message to the verifier which acts as a proof. The improvement of [MY21] over [CVZ19] is that the protocol is dual-mode and has two modes: (1) a proof (with statistical soundness) and computational ZK guarantee and (2) an argument (with computational soundness guarantee) and statistical ZK, rather than only an argument mode, that [CVZ19] enables.

Bartusek, Coladangelo, Khurana and Ma also construct an MDV-NIZK protocol for QMA [BCKM20], and improve our result in two aspects. First, in order to get adaptive soundness (over standard soundness) we need to assume the subexponential (quantum) hardness of LWE, while [BCKM20] only requires assuming the polynomial (quantum) hardness of LWE. Second, our protocol requires polynomially-many copies of the quantum witness for the QMA instance x, while the protocol of [BCKM20] is a single-witness protocol.

2 Preliminaries

We rely on standard notions of classical Turing machines and Boolean circuits:

- A PPT algorithm is a probabilistic polynomial-time Turing machine.
- Let M be a PPT and let x denote the random variable which is the output of M. Whenver the entropy of the output of M is non-zero, we denote the random experiment of sampling x with $x \leftarrow M(\cdot)$. If the entropy of the output of M is zero (i.e. M is deterministic), we denote $x = M(\cdot)$.
- We sometimes think about PPT algorithms as polynomial-size uniform families of circuits, these are equivalent models. A polynomial-size circuit family $\mathcal{C}$ is a sequence of circuits $\mathcal{C} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$, such that each circuit C_λ is of polynomial size $\lambda^{O(1)}$. We say that the family is uniform if there exists a deterministic polynomial-time algorithm M that on input 1^λ outputs C_λ.
- For a PPT algorithm M, we denote by $M(x; r)$ the output of M on input x and random coins r. For such an algorithm and any input x, we write $m \in M(x)$ to denote the fact that m is in the support of $M(x; \cdot)$.

We follow standard notions from quantum computation.

- A QPT algorithm is a quantum polynomial-time Turing machine.

- We sometimes think about QPT algorithms as polynomial-size uniform families of quantum circuits, these are equivalent models. A polynomial-size quantum circuit family $\mathcal{C}$ is a sequence of quantum circuits $\mathcal{C} = \{C_\lambda\}_{\lambda \in \mathbb{N}}$, such that each circuit C_λ is of polynomial size $\lambda^{O(1)}$. We say that the family is uniform if there exists a deterministic polynomial-time algorithm M that on input 1^λ outputs C_λ.
- An interactive algorithm M, in a two-party setting, has input divided into two registers and output divided into two registers. For the input, one register I_m is for an input message from the other party, and a second register I_a is an auxiliary input that acts as an inner state of the party. For the output, one register O_m is for a message to be sent to the other party, and another register O_a is again for auxiliary output that acts again as an inner state. For a quantum interactive algorithm M, both input and output registers are quantum.

The Adversarial Model. Throughout, efficient adversaries are modeled as quantum circuits with non-uniform quantum advice (i.e. quantum auxiliary input). Formally, *a polynomial-size adversary* $\mathsf{A}^* = \{\mathsf{A}_\lambda^*, \rho_\lambda\}_{\lambda \in \mathbb{N}}$, consists of a polynomial-size non-uniform sequence of quantum circuits $\{\mathsf{A}_\lambda^*\}_{\lambda \in \mathbb{N}}$, and a sequence of polynomial-size mixed quantum states $\{\rho_\lambda\}_{\lambda \in \mathbb{N}}$.

For an interactive quantum adversary in a classical protocol, it can be assumed without loss of generality that its output message register is always measured in the computational basis at the end of computation. This assumption is indeed without the loss of generality, because whenever a quantum state is sent through a classical channel then qubits decohere and are effectively measured in the computational basis.

Indistinguishability in the Quantum Setting.

- Let $f : \mathbb{N} \to [0, 1]$ be a function.
  - f is negligible if for every constant $c \in \mathbb{N}$ there exists $N \in \mathbb{N}$ such that for all $n > N$, $f(n) < n^{-c}$.
  - f is noticeable if there exists $c \in \mathbb{N}, N \in \mathbb{N}$ such that for every $n \geq N$, $f(n) \geq n^{-c}$.
  - f is overwhelming if it is of the form $1 - \mu(n)$, for a negligible function μ.
- We may consider random variables over bit strings or over quantum states. This will be clear from the context.
- For two random variables X and Y supported on quantum states, quantum distinguisher circuit D with, quantum auxiliary input ρ, and $\mu \in [0, 1]$, we write $X \approx_{\mathsf{D},\rho,\mu} Y$ if

$$|\Pr[\mathsf{D}(X; \rho) = 1] - \Pr[\mathsf{D}(Y; \rho) = 1]| \leq \mu.$$

- Two ensembles of random variables $\mathcal{X} = \{X_i\}_{\lambda \in \mathbb{N}, i \in I_\lambda}, \mathcal{Y} = \{Y_i\}_{\lambda \in \mathbb{N}, i \in I_\lambda}$ over the same set of indices $I = \cup_{\lambda \in \mathbb{N}} I_\lambda$ are said to be *computationally indistinguishable*, denoted by $\mathcal{X} \approx_c \mathcal{Y}$, if for every polynomial-size quantum distinguisher $\mathsf{D} = \{\mathsf{D}_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ there exists a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}, i \in I_\lambda$,

$$X_i \approx_{\mathsf{D}_\lambda, \rho_\lambda, \mu(\lambda)} Y_i \ .$$

11

– The trace distance between two distributions X, Y supported over quantum states, denoted $\mathrm{TD}(X, Y)$, is a generalization of statistical distance to the quantum setting and represents the maximal distinguishing advantage between two distributions supported over quantum states, by unbounded quantum algorithms. We thus say that ensembles $\mathcal{X} = \{X_i\}_{\lambda \in \mathbb{N}, i \in I_\lambda}$, $\mathcal{Y} = \{Y_i\}_{\lambda \in \mathbb{N}, i \in I_\lambda}$, supported over quantum states, are statistically indistinguishable (and write $\mathcal{X} \approx_s \mathcal{Y}$), if there exists a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}, i \in I_\lambda$,

$$\mathrm{TD}\left(X_i, Y_i\right) \leq \mu(\lambda) \ .$$

In what follows, we introduce the cryptographic tools used in this work. By default, all algorithms are classical and efficient, and security holds against polynomial-size non-uniform quantum adversaries with quantum advice.

2.1 Cryptographic Tools

Interactive Proofs and Sigma Protocols We define interactive proof systems and then proceed to describe sigma protocols, which are a special case of interactive proof systems. In what follows, we denote by (P, V) a protocol between two parties P and V. For common input x, we denote by $\mathsf{OUT}_\mathsf{V}\langle \mathsf{P}, \mathsf{V}\rangle(x)$ the output of V in the protocol. For honest verifiers, this output will be a single bit indicating acceptance or rejection of the proof. Malicious quantum verifiers may have arbitrary quantum output.

Definition 1 (Quantum Proof Systems for QMA). *Let* (P, V) *be a quantum protocol with an honest QPT prover* P *and an honest QPT verifier* V *for a problem* $\mathcal{L} \in \textbf{QMA}$, *satisfying:*

1. **Statistical Completeness:** *There is a polynomial $k(\cdot)$ and a negligible function $\mu(\cdot)$ s.t. for any $\lambda \in \mathbb{N}$, $x \in \mathcal{L} \cap \{0, 1\}^\lambda$, $|w\rangle \in \mathcal{R}_\mathcal{L}(x)$[5],*

$$\Pr[\mathsf{OUT}_\mathsf{V}\langle \mathsf{P}(|w\rangle^{\otimes k(\lambda)}), \mathsf{V}\rangle(x) = 1] \geq 1 - \mu(\lambda) \ .$$

2. **Statistical Soundness:** *There exists a negligible function $\mu(\cdot)$, such that for any (unbounded) prover P^*, any security parameter $\lambda \in \mathbb{N}$, and any $x \in \{0, 1\}^\lambda \setminus \mathcal{L}$,*

$$\Pr\left[\mathsf{OUT}_\mathsf{V}\langle \mathsf{P}^*, \mathsf{V}\rangle(x) = 1\right] \leq \mu(\lambda) \ .$$

We use the abstraction of *Sigma Protocols*, which are public-coin three-message proof systems with a weak zero-knowledge guarantee. We define quantum Sigma Protocols for gap problems in QMA.

Definition 2 (Quantum Sigma Protocol for QMA). *A quantum sigma protocol for* $\mathcal{L} \in \textbf{QMA}$ *is a quantum proof system* $(\Xi.\mathsf{P}, \Xi.\mathsf{V})$ *(as in Definition 1) with 3 messages and the following syntax.*

---

[5] For a problem $\mathcal{L} = (\mathcal{L}_{yes}, \mathcal{L}_{no})$ in QMA, for an instance $x \in \mathcal{L}_{yes}$, the set $\mathcal{R}_\mathcal{L}(x)$ is the (possibly infinite) set of quantum witnesses that make the BQP verification machine accept with some overwhelming probability $1 - \mathrm{negl}(\lambda)$.

- $\alpha = \Xi.\mathsf{P}(|w\rangle^{\otimes k(\lambda)}; r)$: *Given $k(\lambda)$ copies of the quantum witness $w \in \mathcal{R}_{\mathcal{L}}(x)$ and classical randomness r, the first prover message consists of a quantum message α generated by a quantum unitary computation $\Xi.\mathsf{P}$.*
- $\beta \leftarrow \Xi.\mathsf{V}(x)$: *The verifier simply outputs a string of $\mathrm{poly}(|x|)$ random bits.*
- $\gamma = \Xi.\mathsf{P}_3(\beta, r)$: *Given the verifier's β and the randomness r, the prover outputs a response γ by a classical computation $\Xi.\mathsf{P}_3$.*

The protocol satisfies the following.

Special Zero-Knowledge: *There exists a QPT simulator $\Xi.\mathsf{Sim}$ such that,*

$$\left\{ (\alpha, \gamma) \mid r \leftarrow U_{\ell(\lambda)}, \alpha = \Xi.\mathsf{P}(|w\rangle^{\otimes k(\lambda)}; r), \gamma = \Xi.\mathsf{P}_3(\beta, r) \right\}_{\lambda, x, |w\rangle, \beta}$$

$$\approx_c \left\{ (\alpha, \gamma) \mid (\alpha, \gamma) \leftarrow \Xi.\mathsf{Sim}(x, \beta) \right\}_{\lambda, x, |w\rangle, \beta} ,$$

where $\lambda \in \mathbb{N}$, $x \in \mathcal{L} \cap \{0,1\}^{\lambda}$, $|w\rangle \in \mathcal{R}_{\mathcal{L}}(x)$, $\beta \in \{0,1\}^{\mathrm{poly}(\lambda)}$ and $\ell(\lambda)$ is the amount of randomness needed for the first prover message.

Instantiations. Quantum sigma protocols follow from the parallel repetition of the 3-message quantum zero-knowledge protocols of [BG19] for QMA.

Leveled Fully-Homomorphic Encryption with Circuit Privacy We define a leveled fully-homomorphic encryption scheme with circuit privacy, that is, for an encryption $\mathsf{ct} = \mathsf{FHE.Enc}(x)$ and a circuit C, a C-homomorphically-evaluated ciphertext $\hat{\mathsf{ct}} = \mathsf{FHE.Eval}(C, \mathsf{ct})$ reveals nothing on C but $C(x)$.

Definition 3 (Circuit-Private Fully-Homomorphic Encryption). *A circuit-private, leveled fully-homomoprhic encryption scheme* $(\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ *has the following syntax:*

- $\mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda, 1^{s(\lambda)})$: *a probabilistic algorithm that takes a security parameter 1^λ and a circuit size bound $s(\lambda)$ and outputs a secret key sk.*
- $\mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(x)$: *a probabilistic algorithm that given the secret key, takes a string $x \in \{0,1\}^*$ and outputs a ciphertext ct.*
- $\hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, \mathsf{ct})$: *a probabilistic algorithm that takes a (classical) circuit C and a ciphertext ct and outputs an evaluated ciphertext $\hat{\mathsf{ct}}$.*
- $\hat{x} = \mathsf{FHE.Dec}_{\mathsf{sk}}(\hat{\mathsf{ct}})$: *a deterministic algorithm that takes a ciphertext $\hat{\mathsf{ct}}$ and outputs a string $\hat{x}$.*

The scheme satisfies the following.

- **Perfect Correctness:** *For any polynomial $s(\cdot)$, for any $\lambda \in \mathbb{N}$, size-$s(\lambda)$ classical circuit C and input x for C,*

$$\Pr\left[\mathsf{FHE.Dec}_{\mathsf{sk}}(\hat{\mathsf{ct}}) = C(x) \;\middle|\; \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda, 1^{s(\lambda)}), \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc}_{\mathsf{sk}}(x), \\ \hat{\mathsf{ct}} \leftarrow \mathsf{FHE.Eval}(C, \mathsf{ct}) \end{array} \right] = 1 .$$

- **Input Privacy:** *For every polynomial $\ell(\cdot)$ (and any polynomial $s(\lambda)$),*

$$\left\{\mathsf{ct} \ \middle| \ \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda, 1^{s(\lambda)}), \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc_{sk}}(x_0) \end{array}\right\}_{\lambda, x_0, x_1} \approx_c \left\{\mathsf{ct} \ \middle| \ \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{FHE.Gen}(1^\lambda, 1^{s(\lambda)}), \\ \mathsf{ct} \leftarrow \mathsf{FHE.Enc_{sk}}(x_1) \end{array}\right\}_{\lambda, x_0, x_1} ,$$

  where $\lambda \in \mathbb{N}$ and $x_0, x_1 \in \{0,1\}^{\ell(\lambda)}$.
- **Statistical Circuit Privacy:** *There exist unbounded algorithms, probabilistic* Sim *and deterministic* Ext *such that:*
    - *For every $x \in \{0,1\}^*$, $\mathsf{ct} \in \mathsf{FHE.Enc}(x)$, the extractor outputs $\mathsf{Ext}(\mathsf{ct}) = x$.*
    - *For any polynomial $s(\cdot)$,*

$$\{\mathsf{FHE.Eval}(C, \mathsf{ct}^*)\}_{\lambda, C, \mathsf{ct}^*} \approx_s \{\mathsf{Sim}(\ 1^\lambda, C(\mathsf{Ext}(1^\lambda, \mathsf{ct}^*))\ )\}_{\lambda, C, \mathsf{ct}^*} ,$$

  where $\lambda \in \mathbb{N}$, C is a $s(\lambda)$-size circuit, and $\mathsf{ct}^* \in \{0,1\}^*$.

The next claim follows directly from the circuit privacy property, and will be used throughout the analysis.

Claim (Evaluations of Agreeing Circuits are Statistically Close). For any polynomial $s(\cdot)$,

$$\{\mathsf{FHE.Eval}(C_0, \mathsf{ct}^*)\}_{\lambda, C_0, C_1, \mathsf{ct}} \approx_s \{\mathsf{FHE.Eval}(C_1, \mathsf{ct}^*)\}_{\lambda, C_0, C_1, \mathsf{ct}} ,$$

where $\lambda \in \mathbb{N}$, C_0, C_1 are two $s(\lambda)$-size functionally-equivalent circuits, and $\mathsf{ct}^* \in \{0,1\}^*$.

Instantiations. Circuit-private leveled FHE schemes are known based on LWE [OPCPC14, BD18].

Pseudorandom-key Public-key Encryption We define a public-key encryption scheme with pseudorandom public keys.

Definition 4 (Pseudorandom-key Public-key Encryption). *A pseudorandom-key public-key encryption scheme* (PKE.Gen, PKE.Enc, PKE.Dec) *has the following syntax:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$: *a probabilistic algorithm that takes a security parameter 1^λ and outputs a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$.*
- $\mathsf{ct} \leftarrow \mathsf{PKE.Enc_{pk}}(x)$: *a probabilistic algorithm that given the public key, takes a string $x \in \{0,1\}^*$ and outputs a ciphertext ct.*
- $x = \mathsf{PKE.Dec_{sk}}(\mathsf{ct})$: *a deterministic algorithm that given the secret key, takes a ciphertext ct and outputs a string x.*

The scheme satisfies the following.

- **Statistical Correctness Against Malicious Encryptors:** *There is a negligible function $\mathrm{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$ and input $x \in \{0,1\}^*$, the following perfect correctness holds with probability at least $1 - \mathrm{negl}(\lambda)$ over sampling $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$:*

$$\Pr\left[\mathsf{PKE.Dec_{sk}}(\mathsf{ct}) = x \mid \mathsf{ct} \leftarrow \mathsf{PKE.Enc_{pk}}(x)\right] = 1 .$$

– **Public-key Pseudorandomness:** *For $\lambda \in \mathbb{N}$ let $\ell(\lambda)$ be the length of the public key generated by* $\mathsf{PKE.Gen}(1^\lambda)$, *then,*

$$\left\{ \mathsf{pk} \mid (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda) \right\}_{\lambda \in \mathbb{N}} \approx_c \left\{ U_{\ell(\lambda)} \right\}_{\lambda \in \mathbb{N}} \quad .$$

– **Encryption Security:** *For every polynomial $l(\cdot)$,*

$$\left\{ (\mathsf{pk}, \mathsf{ct}) \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda), \\ \mathsf{ct} \leftarrow \mathsf{PKE.Enc}_{\mathsf{pk}}(x_0) \end{array} \right\}_{\lambda, x_0, x_1} \approx_c \left\{ (\mathsf{pk}, \mathsf{ct}) \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda), \\ \mathsf{ct} \leftarrow \mathsf{PKE.Enc}_{\mathsf{pk}}(x_1) \end{array} \right\}_{\lambda, x_0, x_1} ,$$

where $\lambda \in \mathbb{N}$ and $x_0, x_1 \in \{0, 1\}^{l(\lambda)}$.

Instantiations. Pseudorandom-key public-key encryption schemes are known based on LWE [Reg09].

Pseudorandom Function

Definition 5 (Pseudorandom Function (PRF)). *A pseudorandom function scheme* $(\mathsf{PRF.Gen}, \mathsf{PRF.F})$ *has the following syntax:*

– $\mathsf{sk} \leftarrow \mathsf{PRF.Gen}(1^\lambda, 1^{\ell(\lambda)})$: *a probabilistic algorithm that takes a security parameter 1^λ and an output size $\ell(\lambda)$ and outputs a secret key* sk.
– $y = \mathsf{PRF.F}_{\mathsf{sk}}(x)$: *a deterministic algorithm that given the secret key, takes a string $x \in \{0, 1\}^*$ and outputs a string $y \in \{0, 1\}^{\ell(\lambda)}$.*

The scheme satisfies the following property.

– **Pseudorandomness:** *For every quantum polynomial-size distinguisher $\mathsf{D} = \{\mathsf{D}_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ and polynomial $\ell(\cdot)$ there is a negligible function $\mu(\cdot)$ such that for all $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{\mathsf{sk} \leftarrow \mathsf{PRF.Gen}(1^\lambda, 1^{\ell(\lambda)})}[\mathsf{D}_\lambda(\rho_\lambda)^{\mathsf{PRF.F}_{\mathsf{sk}}(\cdot)} = 1] - \Pr_{f \leftarrow (\{0,1\}^{\ell(\lambda)})^{(\{0,1\}^*)}}[\mathsf{D}_\lambda(\rho_\lambda)^{f(\cdot)} = 1] \right| \leq \mu(\lambda) \quad .$$

NIZK Argument for NP in the Common Random String Model We define non-interactive computational zero-knowledge arguments for NP in the common random string model, with adaptive multi-theorem security.

Definition 6 (NICZK Argument for NP). *A non-interactive computational zero-knowledge argument system in the common random string model for a language $\mathcal{L} \in \textbf{NP}$ consists of 3 algorithms* $(\mathsf{NIZK.Setup}, \mathsf{NIZK.P}, \mathsf{NIZK.V})$ *with the following syntax:*

– $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$: *A classical algorithm that on input security parameter λ simply samples a common uniformly random string* crs.
– $\pi \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, x, w)$: *A probabilistic algorithm that on input* crs, *an instance $x \in \mathcal{L}$ and a witness $w \in \mathcal{R}_{\mathcal{L}}(x)$, outputs a proof π.*
– $\mathsf{NIZK.V}(\mathsf{crs}, x, \pi) \in \{0, 1\}$: *A deterministic algorithm that on input* crs, *an instance $x \in \mathcal{L}$ and a proof π, outputs a bit.*

The protocol satisfies the following properties.

- **Perfect Completeness:** *For any $\lambda \in \mathbb{N}$, $x \in \mathcal{L} \cap \{0,1\}^\lambda$, $w \in \mathcal{R}_\mathcal{L}(x)$,*

$$\Pr_{\substack{\mathsf{crs}\leftarrow\mathsf{NIZK.Setup}(1^\lambda),\\ \pi\leftarrow\mathsf{NIZK.P}(\mathsf{crs},x,w)}} \Big[\mathsf{NIZK.V}(\mathsf{crs},x,\pi)=1\Big] = 1 \ .$$

- **Adaptive Computational Soundness:** *For every quantum polynomial-size prover* $\mathsf{NIZK.P}^* = \{\mathsf{NIZK.P}^*_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ *there is a negligible function $\mu(\cdot)$ such that for every security parameter $\lambda \in \mathbb{N}$,*

$$\Pr_{\substack{\mathsf{crs}\leftarrow\mathsf{NIZK.Setup}(1^\lambda),\\ (x,\pi^*)\leftarrow\mathsf{NIZK.P}^*_\lambda(\rho_\lambda,\mathsf{crs})}} \Big[(x \notin \mathcal{L}) \wedge \big(1 = \mathsf{NIZK.V}(\mathsf{crs},x,\pi^*)\big)\Big] \leq \mu(\lambda) \ .$$

- **Multi-Theorem Adaptive Computational Zero Knowledge:** *There exists a polynomial-time simulator* $\mathsf{NIZK.Sim}$ *such that for every quantum polynomial-size distinguisher* $\mathsf{D}^* = \{\mathsf{D}^*_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ *there is a negligible function $\mu(\cdot)$ such that for every security parameter $\lambda \in \mathbb{N}$,*

$$\big|P_{\lambda,\mathsf{Real}} - P_{\lambda,\mathsf{Simulated}}\big| \leq \mu(\lambda) \ ,$$

where,

$$P_{\lambda,\mathsf{Real}} := \Pr_{\mathsf{crs}\leftarrow\mathsf{NIZK.Setup}(1^\lambda)} \Big[\mathsf{D}^*_\lambda(\rho_\lambda,\mathsf{crs})^{\mathsf{NIZK.P}(\mathsf{crs},\cdot,\cdot)} = 1\Big] \ ,$$

$$P_{\lambda,\mathsf{Simulated}} := \Pr_{(\tilde{\mathsf{crs}},\mathsf{td})\leftarrow\mathsf{NIZK.Sim}(1^\lambda)} \Big[\mathsf{D}^*_\lambda(\rho_\lambda,\tilde{\mathsf{crs}})^{\mathsf{NIZK.Sim}(\mathsf{td},\cdot)} = 1\Big] \ ,$$

where,
- *In every query that D^* makes to the oracle, it sends a pair (x,w) where $x \in \mathcal{L} \cap \{0,1\}^\lambda$ and $w \in \mathcal{R}_\mathcal{L}(x)$.*
- $\mathsf{NIZK.P}(\mathsf{crs},\cdot,\cdot)$ *is the prover algorithm and* $\mathsf{NIZK.Sim}(\cdot,\cdot)$ *acts only on its sampled trapdoor* td *and on x.*

Instantiations. Non-interactive computational zero-knowledge arguments for NP in the common random string model with both adaptive soundness and zero knowledge are known based on LWE [CCH$^+$19, PS19].

Malicious Designated-Verifier Non-interactive Zero-knowledge for QMA We define non-interactive zero-knowledge protocols in the malicious designated-verifier model (MDV-NIZK) for QMA, with adaptive (and non-adaptive) multi-theorem security.

Definition 7 (MDV-NICZK Argument for QMA). *A non-interactive computational zero-knowledge argument system for in the malicious designated-verifier model for a gap problem $(\mathcal{L}_{yes}, \mathcal{L}_{no}) = \mathcal{L} \in$ **QMA** consists of 4 algorithms* $(\mathsf{Setup}, \mathsf{VSetup}, \mathsf{P}, \mathsf{V})$ *with the following syntax:*

- $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$: *A classical algorithm that on input security parameter λ simply samples a common uniformly random string* crs.

- $(\mathsf{pvk}, \mathsf{svk}) \leftarrow \mathsf{VSetup}(\mathsf{crs})$: *A classical algorithm that on input* crs *samples a pair of public and secret verification keys.*
- $|\pi\rangle \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{pvk}, x, |w\rangle^{\otimes k(\lambda)})$: *A quantum algorithm that on input* crs*, the public verification key* pvk*, an instance* $x \in \mathcal{L}_{yes}$ *and polynomially-many identical copies of a witness* $|w\rangle \in \mathcal{R}_\mathcal{L}(x)$ *($k(\cdot)$ is some polynomial), outputs a quantum state* $|\pi\rangle$*.*
- $\mathsf{V}(\mathsf{crs}, \mathsf{svk}, x, |\pi\rangle) \in \{0, 1\}$: *A quantum algorithm that on input* crs*, secret verification key* svk*, an instance* $x \in \mathcal{L}$ *and a quantum proof* $|\pi\rangle$*, outputs a bit.*

The protocol satisfies the following properties.

- **Statistical Completeness:** *There is a polynomial* $k(\cdot)$ *and a negligible function* $\mu(\cdot)$ *s.t. for any* $\lambda \in \mathbb{N}$*,* $x \in \mathcal{L}_{yes} \cap \{0,1\}^\lambda$*,* $|w\rangle \in \mathcal{R}_\mathcal{L}(x)$*,* $\mathsf{crs} \in \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pvk}, \mathsf{svk}) \in \mathsf{VSetup}(\mathsf{crs})$*,*

$$\Pr_{|\pi\rangle \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{pvk}, x, |w\rangle^{\otimes k(\lambda)})} \left[\mathsf{V}(\mathsf{crs}, \mathsf{svk}, x, |\pi\rangle) = 1 \right] \geq 1 - \mu(\lambda) \ .$$

- **Multi-Theorem Adaptive Computational Soundness:** *For every quantum polynomial-size prover* $\mathsf{P}^* = \{\mathsf{P}^*_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ *there is a negligible function* $\mu(\cdot)$ *such that for every security parameter* $\lambda \in \mathbb{N}$*,*

$$\Pr_{\substack{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{pvk}, \mathsf{svk}) \leftarrow \mathsf{VSetup}(\mathsf{crs}), \\ (x, |\pi^*\rangle) \leftarrow \mathsf{P}^*_\lambda(\rho_\lambda, \mathsf{crs}, \mathsf{pvk})^{\mathsf{V}(\mathsf{crs}, \mathsf{svk}, \cdot, \cdot)}}} \left[(x \in \mathcal{L}_{no}) \wedge \left(1 = \mathsf{V}(\mathsf{crs}, \mathsf{svk}, x, |\pi^*\rangle) \right) \right] \leq \mu(\lambda) \ .$$

- **Multi-Theorem Adaptive Computational Zero Knowledge:** *There exists a quantum polynomial-time simulator* Sim *such that for every quantum polynomial-size distinguisher* $\mathsf{D}^* = \{\mathsf{D}^*_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ *there is a negligible function* $\mu(\cdot)$ *such that for every security parameter* $\lambda \in \mathbb{N}$*,*

$$\left| \Pr_{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)} \left[\mathsf{D}^*_\lambda(\rho_\lambda, \mathsf{crs})^{\mathsf{P}(\mathsf{crs}, \cdot, \cdot, \cdot)} = 1 \right] - \Pr_{(\tilde{\mathsf{crs}}, \mathsf{td}) \leftarrow \mathsf{Sim}(1^\lambda)} \left[\mathsf{D}^*_\lambda(\rho_\lambda, \tilde{\mathsf{crs}})^{\mathsf{Sim}(\mathsf{td}, \cdot, \cdot)} = 1 \right] \right| \leq \mu(\lambda) \ ,$$

  where,
  - *In every query that* D^* *makes to the oracle, it sends a triplet* $(\mathsf{pvk}^*, x, |w\rangle^{\otimes k(\lambda)})$ *where* pvk^* *can be arbitrary,* $x \in \mathcal{L}_{yes} \cap \{0,1\}^\lambda$ *and* $|w\rangle \in \mathcal{R}_\mathcal{L}(x)$*.*
  - $\mathsf{P}(\mathsf{crs}, \cdot, \cdot, \cdot)$ *is the prover algorithm and* $\mathsf{Sim}(\cdot, \cdot)$ *acts only on its sampled trapdoor* td *and on* pvk^*, x*.*

We note that the standard (non-adaptive) soundness guarantees the following:

Definition 8 (MDV-NICZK Argument for QMA with Standard Soundness). *A non-interactive computational zero-knowledge argument system in the malicious designated-verifier model for a gap problem* $(\mathcal{L}_{yes}, \mathcal{L}_{no}) = \mathcal{L} \in \mathbf{QMA}$ *has standard non-adaptive soundness if it satisfies the same properties described in definition 7, with the only change that instead of satisfying multi-theorem adaptive soundness, it satisfies the following guarantee:*

17

– **Multi-Theorem Computational Soundness:** *For every quantum polynomial-size prover* $\mathsf{P}^* = \{\mathsf{P}^*_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ *and* $\{x_\lambda\}_{\lambda \in \mathbb{N}}$ *where* $\forall \lambda \in \mathbb{N} : x_\lambda \in \mathcal{L}_{no}$*, there is a negligible function* $\mu(\cdot)$ *such that for every security parameter* $\lambda \in \mathbb{N}$*,*

$$\Pr_{\substack{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{pvk}, \mathsf{svk}) \leftarrow \mathsf{VSetup}(\mathsf{crs}), \\ |\pi^*\rangle \leftarrow \mathsf{P}^*_\lambda(\rho_\lambda, \mathsf{crs}, \mathsf{pvk})^{\mathsf{V}(\mathsf{crs}, \mathsf{svk}, \cdot, \cdot)}}} \left[1 = \mathsf{V}(\mathsf{crs}, \mathsf{svk}, x, |\pi^*\rangle) \right] \leq \mu(\lambda) \ .$$

3 Non-interactive Zero-knowledge Protocol

In this section we describe a non-interactive computational zero-knowledge argument system in the malicious designated-verifier model for an arbitrary $\mathcal{L} \in \mathbf{QMA}$, according to Definition 7.

Ingredients and notation:

– A non-interactive zero-knowledge argument for NP $(\mathsf{NIZK.Setup}, \mathsf{NIZK.P}, \mathsf{NIZK.V})$ in the common random string model.
– A pseudorandom function $(\mathsf{PRF.Gen}, \mathsf{PRF.F})$.
– A leveled fully-homomorphic encryption scheme $(\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ with circuit privacy.
– A public-key encryption scheme $(\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with pseudorandom public keys.
– A 3-message quantum sigma protocol $(\varXi.\mathsf{P}, \varXi.\mathsf{V})$ for QMA.

We describe the protocol in Figure 1.

The (statistical) completeness of the protocol follows readily from the perfect completeness of the NIZK scheme, the perfect correctness of FHE and the statistical completeness of the quantum sigma protocol $(\varXi.\mathsf{P}, \varXi.\mathsf{V})$. We next prove the soundness and zero knowledge of the protocol.

3.1 Soundness

We prove that the protocol has multi-theorem computational soundness (as in Definition 8). By standard generic compilation and sub-exponential hardness of LWE we extend our soundness to be adaptive (as in Definition 7).

Proposition 1 (The Protocol has Multi-theorem Computational Soundness). *For every quantum polynomial-size prover* $\mathsf{P}^* = \{\mathsf{P}^*_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ *there is a negligible function* $\mu(\cdot)$ *such that for every security parameter* $\lambda \in \mathbb{N}$ *and* $x \in \mathcal{L}_{no} \cap \{0,1\}^\lambda$*,*

$$\Pr_{\substack{(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \left((\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V}), (\mathsf{prfk}, \mathsf{fhek})\right) \leftarrow \mathsf{VSetup}(\mathsf{crs}, \mathsf{ek}), \\ |\pi^*\rangle \leftarrow \mathsf{P}^*_\lambda\left(\rho_\lambda, (\mathsf{crs}, \mathsf{ek}), (\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V})\right)^{\mathsf{V}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), \cdot, \cdot)}}} \left[1 = \mathsf{V}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), x, |\pi^*\rangle) \right] \leq \mu(\lambda) \ .$$

Protocol 1

Common Input: An instance $x \in \mathcal{L}_{yes} \cap \{0,1\}^\lambda$, for security parameter $\lambda \in \mathbb{N}$.

P's private input: Polynomially many identical copies of a witness for x: $|w\rangle^{\otimes k(\lambda)}$ s.t. $|w\rangle \in \mathcal{R}_{\mathcal{L}}(x)$.

1. **Common Random String:** Setup samples the common random string of the NP NIZK argument, $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$ and an additional random string $\mathsf{ek} \leftarrow U_{\ell(\lambda)}$ where $\ell(\lambda)$ is the size of a public key generated by $\mathsf{PKE.Gen}(1^\lambda)$. Setup publishes $(\mathsf{crs}, \mathsf{ek})$ as the common random string.

2. **Public and Secret Verification Keys:** VSetup samples public and secret verification keys:
   - Samples $\mathsf{prfk} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, $\mathsf{fhek} \leftarrow \mathsf{FHE.Gen}(1^\lambda)$ and encrypts the PRF key using the FHE encryption, $\mathsf{ct_V} \leftarrow \mathsf{FHE.Enc_{fhek}}(\mathsf{prfk})$.
   - Let r_V be the randomness used for $\mathsf{PRF.Gen}, \mathsf{FHE.Gen}, \mathsf{FHE.Enc}$. VSetup encrypts $\mathsf{ct}_{r_\mathsf{V}} \leftarrow \mathsf{PKE.Enc_{ek}}(r_\mathsf{V})$ and computes a NIZK proof $\pi_\mathsf{V} \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, (\mathsf{ct_V}, \mathsf{ct}_{r_\mathsf{V}}, \mathsf{ek}))$, for the NP statement declaring that the tuple $(\mathsf{ct_V}, \mathsf{ct}_{r_\mathsf{V}}, \mathsf{ek})$ is consistent.[a]

   The key values are: $\mathsf{pvk} = (\mathsf{ct_V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V})$, $\mathsf{svk} = (\mathsf{prfk}, \mathsf{fhek})$.

3. **Non-interactive Zero-knowledge Proof:** Given $(\mathsf{crs}, \mathsf{ek})$ and pvk, P first checks that $1 = \mathsf{NIZK.V}(\mathsf{crs}, (\mathsf{ct_V}, \mathsf{ct}_{r_\mathsf{V}}, \mathsf{ek}), \pi_\mathsf{V})$ and aborts otherwise.
   - P computes the sigma protocol message $\alpha = \Xi.\mathsf{P}(|w\rangle^{\otimes k(\lambda)}; r_\Xi)$, for randomness r_Ξ.
   - P computes $\hat{\mathsf{ct}}_\mathsf{P} \leftarrow \mathsf{FHE.Eval}(C_{x,r_\Xi}, \mathsf{ct_V})$, where C_{x,r_Ξ} is the following circuit: Given input prfk a PRF secret key, C_{x,r_Ξ} computes $\beta_x = \mathsf{PRF.F_{prfk}}(x)$, and then outputs $\gamma = \Xi.\mathsf{P}_3(\beta_x, r_\Xi)$.
   - P encrypts $\mathsf{ct}_{r_\Xi} \leftarrow \mathsf{PKE.Enc_{ek}}(r_\Xi)$ and computes a NIZK proof $\pi_\mathsf{P} \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, (\hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \mathsf{ek}))$, for the NP statement declaring that the tuple $(\hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \mathsf{ek})$ is consistent.[b]

   P sends $|\pi\rangle = (\alpha, \hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \pi_\mathsf{P})$ to V.

4. **Verification:** Given $(\mathsf{crs}, \mathsf{ek})$, svk and $|\pi\rangle$, V accepts iff all of the following holds:
   - $1 = \mathsf{NIZK.V}(\mathsf{crs}, (\hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \mathsf{ek}), \pi_\mathsf{P})$.
   - Let $\beta_x = \mathsf{PRF.F_{prfk}}(x)$, $\gamma = \mathsf{FHE.Dec_{fhek}}(\hat{\mathsf{ct}}_\mathsf{P})$, then $1 = \Xi.\mathsf{V}(x, \alpha, \beta_x, \gamma)$.

---

[a]Formally, there exist r_1, r_2 s.t. $\mathsf{ct_V}$ is generated by using $\mathsf{PRF.Gen}, \mathsf{FHE.Gen}, \mathsf{FHE.Enc}$ with randomness r_1, and $\mathsf{ct}_{r_\mathsf{V}} = \mathsf{PKE.Enc_{ek}}(r_1; r_2)$.

[b]Formally, there exist r_Ξ, r_1, r_2 s.t. $\hat{\mathsf{ct}}_\mathsf{P} = \mathsf{FHE.Eval}(C_{x,r_\Xi}, \mathsf{ct_V}; r_1)$, $\mathsf{ct}_{r_\Xi} = \mathsf{PKE.Enc_{ek}}(r_\Xi; r_2)$.

Fig. 1: A non-interactive computational zero-knowledge argument system for $\mathcal{L} \in \mathbf{QMA}$ in the malicious designated-verifier model.

Proof. Let $\mathsf{P}^* = \{\mathsf{P}^*_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ a polynomial-size quantum prover and let $\{x_\lambda\}_{\lambda \in \mathbb{N}}$ s.t. $\forall \lambda \in \mathbb{N} : x_\lambda \in \mathcal{L}_{no} \cap \{0,1\}^\lambda$. We prove soundness by a hybrid argument, that is, we consider a series of computationally-indistinguishable hybrid processes with output over $\{0,1\}$, starting from the output of the verifier (for the prover's false proof) in the

real inteaction, until we get to a distribution where the output of the verifier can be 1 with at most negligible probability. We define the following processes.

- Hyb_0 : The output distribution of the verifier in the real interaction, that is, for

$$(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda) \;,\; \big((\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V}), (\mathsf{prfk}, \mathsf{fhek})\big) \leftarrow \mathsf{VSetup}(\mathsf{crs}, \mathsf{ek})\;,$$

$$|\pi^*\rangle \leftarrow \mathsf{P}^*_\lambda\big(\rho_\lambda, (\mathsf{crs}, \mathsf{ek}), (\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V})\big)^{\mathsf{V}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), \cdot, \cdot)}\;,$$

  the output bit $\mathsf{V}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), x, |\pi^*\rangle)$.
- Hyb_1 : This hybrid process is identical to Hyb_0, with the exception that ek is sampled as a public key for the PKE scheme $(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{PKE}.\mathsf{Gen}(1^\lambda)$, rather than as a random string of the same length. To move to this hybrid we will use the fact that the public keys of the PKE scheme are pseudorandom.
- Hyb_2 : This hybrid process is identical to Hyb_1, with the exception that the verification algorithm (described in step 4 of the protocol) changes. The new verifier $\tilde{\mathsf{V}}$ still makes sure that π_P is a valid proof for $(\hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \mathsf{ek})$, but the second check changes to the following: Let $r_\Xi = \mathsf{PKE}.\mathsf{Dec}_\mathsf{sk}(\mathsf{ct}_{r_\Xi})$, and let $\gamma = \Xi.\mathsf{P}_3(\beta_x, r_\Xi)$. Then $\tilde{\mathsf{V}}$ accepts if $1 = \Xi.\mathsf{V}(x, \alpha, \beta_x, \gamma)$. To move to this hybrid we will use the (adaptive) soundness property of the NP NIZK proof that P^* provides.
- Hyb_3 : This hybrid process is identical to Hyb_2, with the exception that when generating the CRS $(\mathsf{crs}, \mathsf{ek})$ and the public verification key $\mathsf{pvk} = (\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V})$, (1) the CRS for the NP NIZK is simulated $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK}.\mathsf{Sim}(1^\lambda)$, (2) the proof π_V is simulated $\pi_\mathsf{V} \leftarrow \mathsf{NIZK}.\mathsf{Sim}(\mathsf{td}, (\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \mathsf{ek}))$ rather than generated by the NP NIZK prover. To move to this hybrid we use the zero-knowledge property of the NP NIZK proof that V provides.
- Hyb_4 : This hybrid process is identical to Hyb_3, with the exception that when generating $\mathsf{pvk} = (\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V})$, $\mathsf{ct}_{r_\mathsf{V}}$ is just an encryption of a string of zeros (of the same length) rather than the randomness r_V. To move to this hybrid we use the security of the PKE scheme.
- Hyb_5 : This hybrid process is identical to Hyb_4, with the exception that when generating $\mathsf{pvk} = (\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V})$, ct_V is just an encryption of a string of zeros (of the same length) rather than the FHE encryption of the secret PRF key prfk. To move to this hybrid we use the security of the FHE scheme.
- Hyb_6 : This hybrid process is identical to Hyb_5, with the exception that the modified verification algorithm $\tilde{\mathsf{V}}$ from Hyb_2 is now going to be a new *stateful* algorithm $\tilde{\mathsf{V}}_s$. The new verifier $\tilde{\mathsf{V}}_s$ still makes sure that π_P is a valid proof for $(\hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \mathsf{ek})$, but the second check changes to the following: It is identical to that of $\tilde{\mathsf{V}}$, except that β_x is now lazily sampled as a truly random string, that is, every time P^* sends a query for some x', instead of computing $\beta_{x'} = \mathsf{PRF}.\mathsf{F}_{\mathsf{prfk}}(x')$, $\tilde{\mathsf{V}}_s$ samples $\beta_{x'}$ a truly random string of the same length and remembers it for future queries by the prover (for the same x'). To move to this hybrid we use the pseudorandomnes guarantee of the PRF.
- Hyb_7 : This hybrid process is identical to Hyb_6, with the exception that the behaviour of the verification algorithm $\tilde{\mathsf{V}}_s$ changes in the following way: Consider t the *first* time step in the execution of P^* (in Hyb_6) such that with a noticeable probability, P^* sends a pair $(x', |\pi^*\rangle)$ such that (1) $x' \in \mathcal{L}_{no}$ and (2) the modified verification

algorithm $\tilde{V}_s$ accepts - this proof can be sent either as a query to the verification oracle, or as the final output of P^* (in that case, t is the last time step of P^* and $x' = x$).

Now we define Hyb_7: the verification algorithm works as in Hyb_6 with the one change that if P^* sends a query to the verification oracle before its time step t and this query is for a no-instance $x' \in \mathcal{L}_{no}$, then we simply return 0 to P^* as the verifier's answer, without computing anything. Note that checking whether $x' \in \mathcal{L}_{no}$ takes $2^{O(|x'|)}$ time[6], and thus the execution of this hybrid is inefficient. If such time step t does not exist (i.e. in each of the prover's time steps, the probability for it to generate a false proof is only negligible), this process is identical to Hyb_6.

We now explain why the outputs of each two consecutive hybrids are computationally indistinguishable[7]. We will then use the last hybrid process to show that soundness of the protocol follows from the soundness of the quantum sigma protocol $(\Xi.\mathsf{P}, \Xi.\mathsf{V})$.

- $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1$: Follows readily from the pseudorandomness property of the public keys generated by $\mathsf{PKE.Gen}(1^\lambda)$.
- $\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2$: Follows from the adaptive soundness of the NIZK protocol for NP, the statistical correctness of the PKE scheme and the perfect correctness of the FHE scheme. We explain in more detail: Assume the output bits of Hyb_1 and Hyb_2 are distinguishable with some noticeable advantage, then by the perfect correctness of the FHE evaluation, it follows that with a noticeable probability, either (1) there was an error in the decryption process of the PKE scheme at least once, or (2) P^* generated a false proof for the NP NIZK scheme at least once. We prove that both happen with at most negligible probability, and thus the statistical distance between the output bits of Hyb_1 and Hyb_2 is at most negligible.

    The correctness guarantee of the PKE scheme is that when the public key is sampled honestly, which is true in our case, then with overwhelming probability over the randomness of $\mathsf{PKE.Gen}(1^\lambda)$, the decryption is perfectly correct, regardless of the randomness used for the encryption (which in our case is possibly malicious, as it is chosen by P^*). This implies that with at most negligible probability there is an error in the decryption process $\mathsf{PKE.Dec}_{\mathsf{sk}}(\cdot)$.

    If P^* manages to give a false proof π_{P}^* for some tuple $(\hat{\mathsf{ct}}_{\mathsf{P}}, \mathsf{ct}_{r_\Xi}, \mathsf{ek})$ with a noticeable probability ε then we can use it to break the adaptive soundness of the NP NIZK scheme: We guess the index of the query (to the verification oracle $\tilde{\mathsf{V}}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), \cdot, \cdot)$) where P^* gives such false proof, and with probability at least $\varepsilon \cdot \frac{1}{t}$, where t is the (polynomial) running time of P^*, we find such false proof. This implies that ε has to be at most negligible i.e. P^* cannot produce a false proof for the NP NIZK with a noticeable probability.

---

[6]We assume that our gap problem $\mathcal{L} \in \mathbf{QMA}$ has exponential-time algorithms that solve it, that is, for $x \in \mathcal{L}$ we can decide whether $x \in \mathcal{L}_{yes}$ or $x \in \mathcal{L}_{no}$ in $2^{O(|x|)}$ time. It is also enough for our proof to assume that $\mathcal{L}$ is solvable in general exponential time i.e. $O(2^{|x|^c})$ time for some constant $c \in \mathbb{N}$.

[7]the output bits of the hybrids are in fact *statistically* indistinguishable, because any two distributions over a bit are statistically indistinguishable if they are computationally indistinguishable, but we won't care about this in our analysis.

- $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$: Assume toward contradiction that the output bits of Hyb_2 and Hyb_3 are distinguishable with some noticeable advantage, we use the prover P^* in order to construct a distinguisher D that breaks the zero-knowledge property of the NP NIZK scheme (it seems that we don't have to use the fact that the zero knowledge property of the NP NIZK is adaptive, but we will use it for the convenience of the proof and because it does not cause an extra cost in computational assumptions). D will sample $(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$, honestly sample $(\mathsf{ct_V}, \mathsf{ct}_{r_V})$ with randomness r, and then get a common random string crs from the NIZK zero knowledge challenger. D then hands $(\mathsf{ct_V}, \mathsf{ct}_{r_V}, \mathsf{ek})$ along with the NP witness r and gets back either a real proof or a simulated proof. it then proceeds to run the malicious prover P^* and at the end, by the verdict of the (modified) verification algorithm $\tilde{\mathsf{V}}$ for the prover's proof and instance, distinguishes between whether it got a simulated proof or a real proof. This follows from the fact that whenever D gets a real proof (and CRS) then the view of P^* is exactly its view in Hyb_2 and whenever D gets a simulated proof (and CRS) then the view of P^* is exactly its view in Hyb_3.
- $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$: Follows readily from the security of the PKE scheme.
- $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$: Follows readily from the security of the FHE scheme.
- $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$: Follows readily from the security of the PRF scheme.
- $\mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7$: Note that by how we defined the time step t it follows that the change of returning 0 on queries for no-instances before time step t (rather than actually evaluating the verification algorithm $\tilde{\mathsf{V}}_s$) is unnoticeable to the prover P^*.

Now, assume toward contradiction that P^* succeeds in breaking the soundness with a noticeable probability in the original execution of the protocol (i.e. in the process Hyb_0), and by the fact $\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_7$ it follows that the verifier accepts the prover's false proof with some noticeable probability in the hybrid experiment Hyb_7. By the fact that with some noticeable probability P^* succeeds in cheating in Hyb_7, it follows that a time step t exists where P^* sends a pair $(x', |\pi^*\rangle)$ such that $x' \in \mathcal{L}_{no}$ and $\tilde{\mathsf{V}}_s$ accepts the proof (this follows because in the last step of P^*'s execution it sends noticeably often a successful false proof for $x \in \mathcal{L}_{no}$).

Now we consider the execution process of Hyb_7 and fix by an averaging argument the snapshot $|\psi\rangle$ of the execution in the exact moment where P^* sends a pair $(x', |\pi^*\rangle)$ in its time step t, such that the snapshot maximizes the probability that $x' \in \mathcal{L}_{no}$ and $\tilde{\mathsf{V}}_s$ accepts the proof $|\pi^*\rangle$ (as a side note, this snapshot includes (1) all of the randomness (including setup information) in the process Hyb_7 until P^*'s step t, (2) the inner quantum state of P^* in step t, and of course a pair $(x', |\pi^*\rangle)$ such that $x' \in \mathcal{L}_{no}$.). It follows that the part α and the extracted γ (both obtained from $|\pi^*\rangle$, recall γ is obtained by the extracted randomness r_Ξ and the random string $\beta_{x'}$) make a quantum sigma protocol verifier $\Xi.\mathsf{V}$ accept the proof for a random challenge β with a noticeable probability.

We now describe a malicious prover $\Xi.\mathsf{P}^*$ that breaks the soundness of the quantum sigma protocol $(\Xi.\mathsf{P}, \Xi.\mathsf{V})$, by using P^* and the quantum advice $|\psi\rangle$ in order to convince $\Xi.\mathsf{V}$ to accept the no-instance $x' \in \mathcal{L}_{no}$. $\Xi.\mathsf{P}^*$ uses the snapshot $|\psi\rangle$ and takes α from $|\pi^*\rangle$ and sends it as the first sigma protocol message to $\Xi.\mathsf{V}$. $\Xi.\mathsf{V}$ returns a random challenge β, and $\Xi.\mathsf{P}^*$ treats this random challenge as the random $\beta_{x'}$ for the verification procedure $\tilde{\mathsf{V}}_s$. $\Xi.\mathsf{P}^*$ then derives γ from $|\pi^*\rangle$ (as usual in $\tilde{\mathsf{V}}_s$) and sends it to $\Xi.\mathsf{V}$. Recall that we know $\Xi.\mathsf{V}$ accepts the proof with a noticeable probability, and thus $\Xi.\mathsf{P}^*$

breaks the soundness of the quantum sigma protocol with noticeable probability, in contradiction.

We next use standard complexity leveraging to make the soundness adaptive, that is, by assuming that the security of our cryptographic primitives is sub-exponential we prove that the prover cannot choose the no-instance $x \in \mathcal{L}_{no}$ adaptively. As mentioned in the preliminaries, the security of all of our primitives can be based on the hardness of LWE, and thus based on the sub-exponential hardness of LWE we can get adaptive soundness.

Proposition 2 (The Protocol has Multi-theorem Adaptive Computational Soundness). *Assume there is a constant $\varepsilon \in (0, 1)$ such that the cryptographic ingridients we use are secure against $O(2^{\lambda^{\varepsilon}})$-time quantum algorithms for security paramter λ. Then, by executing the protocol with security parameter $\lambda := |x|^{\frac{2}{\varepsilon}}$ rather than $\lambda = |x|$, for every quantum polynomial-size prover $\mathsf{P}^* = \{\mathsf{P}^*_{\lambda}, \rho_{\lambda}\}_{\lambda \in \mathbb{N}}$ there is a negligible function $\mu(\cdot)$ such that for every security parameter $\lambda \in \mathbb{N}$,*

$$\Pr\left[(x \in \mathcal{L}_{no}) \wedge \left(1 = \mathsf{V}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), x, |\pi^*\rangle))\right) \right] \leq \mu(\lambda) \ ,$$

where the probability is above the following experiment:

$$(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^{\lambda}), \ \ \left((\mathsf{ct}_{\mathsf{V}}, \mathsf{ct}_{r_{\mathsf{V}}}, \pi_{\mathsf{V}}), (\mathsf{prfk}, \mathsf{fhek})\right) \leftarrow \mathsf{VSetup}(\mathsf{crs}, \mathsf{ek}),$$

$$(x, |\pi^*\rangle) \leftarrow \mathsf{P}^*_{\lambda}\left(\rho_{\lambda}, (\mathsf{crs}, \mathsf{ek}), (\mathsf{ct}_{\mathsf{V}}, \mathsf{ct}_{r_{\mathsf{V}}}, \pi_{\mathsf{V}})\right)^{\mathsf{V}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), \cdot, \cdot)} \ .$$

Proof. The proof is almost identical to the proof of Proposition 1, with minor technical changes. Let $\mathsf{P}^* = \{\mathsf{P}^*_{\lambda}, \rho_{\lambda}\}_{\lambda \in \mathbb{N}}$ a polynomial-size quantum prover in Protocol 1 and as before, we prove soundness by a hybrid argument by considering almost the same series of hybrids processes, and the reductions that show the outputs of each consecutive pair of hybrids are indistinguishable, are also going to be slightly different.

More precisely, consider the exact same hybrids $\mathsf{Hyb}_0, \cdots, \mathsf{Hyb}_7$ from the proof of Proposition 1, with only the following differences:

- With accordance to the fact that we consider adaptive provers, in each hybrid process, the output of the malicious prover at the end of the execution is a pair $(x, |\pi^*\rangle)$ rather than only a proof $|\pi^*\rangle$.
- The output of each hybrid process is still a bit, but going to be the logical AND of (1) the verifier accepting the prover's proof and instance x, and (2) the instance x is indeed a no-instance $x \in \mathcal{L}_{no}$ (note that in the proof for Proposition 1 the output bit of the hybrids only considers the verdict of the verifier, as the no-instance $x \in \mathcal{L}_{no}$ is already fixed).

We will next claim that the outputs of each pair of consecutive hybrids are computationally indistinguishable. For this, we will use the fact that given $x \in \mathcal{L} = \mathcal{L}_{yes} \cup \mathcal{L}_{no}$, we can decide whether $x \in \mathcal{L}_{no}$ or not in $2^{O(|x|)}$ time.[8] We also use the fact that our

---

[8]As noted before, the proof is not sensitive to the fact that the time complexity is $2^{O(|x|)}$ and not $O(2^{|x|^c})$ time for some constant $c \in \mathbb{N}$.

primitives are assumed to be secure against sub-exponential time algorithms and we run the protocol with increased security parameter, more specifically, we assume that our primitives are secure against $O(2^{\lambda^\varepsilon})$-time algorithms and we use security paramter $\lambda = |x|^{\frac{2}{\varepsilon}}$, thus it follows that no $O(2^{\lambda^\varepsilon}) = O(2^{|x|^2})$-time algorithm can break the security of the primitives.

In continuance to the above, by the exact same reductions from the proof of Proposition 1 with a single change, we have

$$\mathsf{Hyb}_0 \approx_c \mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6 \approx_c \mathsf{Hyb}_7 \ .$$

The single change that we refer to is the check that the reduction makes when getting the final output of the prover. In the proof of Proposition 1, the final output of P^* is a false proof $|\pi^*\rangle$ for a specific and pre-chosen x, while in our case (the adaptive case) it is a pair $(x, |\pi^*\rangle)$ for an adaptively-chosen x. Instead of checking only the verdict of V, which can be done in polynomial time, the reduction in our case will also check that $x \in \mathcal{L}_{no}$, which can be done in time $2^{O(|x|)}$. This implies that our security reductions take $2^{O(|x|)}$ time to execute, but they break primitives with security against $O(2^{|x|^2})$-time algorithms, which constitutes the needed contradiction. Finally, the algorithm $\Xi.\mathsf{P}^*$ that uses P^* in the process Hyb_7 in order to break the soundness of the quantum sigma protocol is exactly the same as before, and our proof is finished.

As mentioned before, by the fact that the security of the cryptographic ingridients in our protocol can be based on the hardness of LWE and the security reductions for the primitives are polynomial-time, we get the following corollary.

Corollary 1. *Assume there is a constant $\varepsilon \in (0, 1)$ such that LWE is hard for $O(2^{n^\varepsilon})$-time quantum algorithms (for LWE secret of n bits). Then, for every quantum polynomial-size prover $\mathsf{P}^* = \{\mathsf{P}^*_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ there is a negligible function $\mu(\cdot)$ such that for every security parameter $\lambda \in \mathbb{N}$,*

$$\Pr\left[(x \in \mathcal{L}_{no}) \wedge \left(1 = \mathsf{V}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), x, |\pi^*\rangle)\right)\right] \leq \mu(\lambda) \ ,$$

where the probability is above the following experiment:

$$(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda), \ \ \left((\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V}), (\mathsf{prfk}, \mathsf{fhek})\right) \leftarrow \mathsf{VSetup}(\mathsf{crs}, \mathsf{ek}),$$

$$(x, |\pi^*\rangle) \leftarrow \mathsf{P}^*_\lambda\left(\rho_\lambda, (\mathsf{crs}, \mathsf{ek}), (\mathsf{ct}_\mathsf{V}, \mathsf{ct}_{r_\mathsf{V}}, \pi_\mathsf{V})\right)^{\mathsf{V}((\mathsf{crs}, \mathsf{ek}), (\mathsf{prfk}, \mathsf{fhek}), \cdot, \cdot)} \ .$$

3.2 Zero Knowledge

We show that the protocol is multi-theorem adaptive computational zero-knowledge[9], which holds even when the trusted setup samples only a common uniformly random

---

[9]It would have been enough to show that the protocol is *single-theorem* adaptive computational zero-knowledge, and then by the single-to-multi-theorem compiler for NIZKs of [FLS99] get a MDV-NICZK argument with adaptive *multi-theorem* security, but for the sake of completeness, because our construction can be shown to be multi-theorem zero-knowledge without the FLS compilation and because it does not change the main ideas in the proof, we prove the multi-theorem case directly.

string, and an adversarial polynomial-time (quantum) verifier samples its public verification key maliciously.

We next describe the simulator and then prove that the view that it generates is indistinguishable from the real one, against adaptive distinguishers that choose the statement to be proven only after seeing the common random string.

$\mathsf{Sim}(1^\lambda)$:

1. **CRS Simulation:** Given a security parameter λ, the first simulator output is the simulation of the CRS for the NP NIZK protocol and swapping ek with a public key for the PKE scheme, that is, Sim samples:

$$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Sim}(1^\lambda) \ , (\mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda) \ ,$$

   outputs $(\mathsf{crs}, \mathsf{ek})$ as the simulated CRS and $(\mathsf{td}, \mathsf{sk})$ as the simulator trapdoor.

2. **Proof Simulation:** Given the trapdoor $(\mathsf{td}, \mathsf{sk})$, a (possibly malicious) public verification key $\mathsf{pvk} = (\mathsf{ct_V}, \mathsf{ct}_{r_V}, \pi_V)$ and a yes-instance $x \in \mathcal{L}_{yes}$, the simulator does the following:

   (a) Sim checks that π_V is a valid proof for the tuple $(\mathsf{ct_V}, \mathsf{ct}_{r_V}, \mathsf{ek})$ and also actually verifies some of the statement itself: It decrypts $r_V = \mathsf{PKE.Dec_{sk}}(\mathsf{ct}_{r_V})$ and checks that $\mathsf{ct_V}$ is obtained by running $\mathsf{PRF.Gen}, \mathsf{fhek} \leftarrow \mathsf{FHE.Gen}, \mathsf{FHE.Enc_{fhek}}$ with randomness r_V. If the check is not accepted, Sim returns $\bot$.

   (b) Sim derives prfk from r_V, computes $\beta_x = \mathsf{PRF.F_{prfk}}(x)$ and then executes $(\alpha, \gamma) \leftarrow \Xi.\mathsf{Sim}(x, \beta_x)$.

   (c) Sim performs a circuit-private homomorphic evaluation $\hat{\mathsf{ct}}_\mathsf{P} \leftarrow \mathsf{FHE.Eval}(C_\gamma, \mathsf{ct_V})$, where C_γ is the circuit that always outputs γ.

   (d) Sim encrypts $\mathsf{ct}_{r_\Xi} \leftarrow \mathsf{PKE.Enc_{ek}}(0^\ell)$, where ℓ is the length of the randomness for the prover in the quantum sigma protocol.

   (e) Finally, Sim simulates the non-interactive zero-knowledge proof π_P, by executing $\pi_\mathsf{P} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}, (\hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \mathsf{ek}))$.

   Sim outputs $(\alpha, \hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \pi_\mathsf{P})$.

We now prove that the simulated proofs that the simulator generates are computationally indistinguishable from the real proofs that the prover generates.

Proposition 3 (The Protocol is Multi-theorem Adaptive Computational Zero-knowledge).
For every quantum polynomial-size distinguisher $\mathsf{D}^* = \{\mathsf{D}^*_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ there is a negligible function $\mu(\cdot)$ such that for every security parameter $\lambda \in \mathbb{N}$,

$$|P_{\lambda,\mathsf{Real}} - P_{\lambda,\mathsf{Simulated}}| \leq \mu(\lambda) \ ,$$

where,

$$P_{\lambda,\mathsf{Real}} := \Pr_{(\mathsf{crs},\mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)} \left[\mathsf{D}^*_\lambda(\rho_\lambda, (\mathsf{crs}, \mathsf{ek}))^{\mathsf{P}((\mathsf{crs},\mathsf{ek}),\cdot,\cdot,\cdot)} = 1 \right] \ ,$$

$$P_{\lambda,\mathsf{Simulated}} := \Pr_{((\mathsf{crs},\mathsf{ek}),(\mathsf{td},\mathsf{sk})) \leftarrow \mathsf{Sim}(1^\lambda)} \left[\mathsf{D}^*_\lambda(\rho_\lambda, (\mathsf{crs}, \mathsf{ek}))^{\mathsf{Sim}((\mathsf{td},\mathsf{sk}),\cdot,\cdot)} = 1 \right] \ ,$$

where in every query that D^* makes to the oracle, it sends a triplet $(\mathsf{pvk}^*, x, |w\rangle^{\otimes k(\lambda)})$ such that pvk^* can be arbitrary, $x \in \mathcal{L}_{yes} \cap \{0,1\}^\lambda$ and $|w\rangle \in \mathcal{R}_\mathcal{L}(x)$.

Proof. Let $\mathsf{D}^* = \{\mathsf{D}^*_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$ a polynomial-size quantum distinguisher. We prove zero knowledge by a hybrid argument, that is, we consider a series of computationally-indistinguishable hybrid processes with 1-bit outputs, starting from the output of D^* when getting real proofs, until we get to the output of D^* when getting simulated proofs. We define the following processes.

- Hyb_0 : The output of D^* when getting honestly-generated proofs, that is, it gets the CRS from $(\mathsf{crs}, \mathsf{ek}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the proofs from $\mathsf{P}^*((\mathsf{crs}, \mathsf{ek}), \cdot, \cdot, \cdot)$, as described in the experiment of P_{Real}.

- Hyb_1 : This hybrid process is identical to Hyb_0, with the exception that ek is sampled as a public key for the PKE scheme $(\mathsf{ek}, \mathsf{sk}) \leftarrow \mathsf{PKE.Gen}(1^\lambda)$, rather than as a random string of the same length. To move to this hybrid we will use the fact that the public keys of the PKE scheme are pseudorandom.

- Hyb_2 : This hybrid process is identical to Hyb_1, with the exception that the prover adds another validity check, over the one checking the validity of the proof π_V: It decrypts $r_\mathsf{V} = \mathsf{PKE.Dec}_{\mathsf{sk}}(\mathsf{ct}_{r_\mathsf{V}})$ and checks that ct_V is obtained by running $\mathsf{PRF.Gen}, \mathsf{fhek} \leftarrow \mathsf{FHE.Gen}, \mathsf{FHE.Enc}_{\mathsf{fhek}}$ with randomness r_V. To move to this hybrid we will use the adaptive soundness of the NP NIZK.

- Hyb_3 : This hybrid process is identical to Hyb_2, with the exception that we simulate the NP NIZK proofs, that is, (1) when sampling the NP NIZK common random string crs from the total CRS $(\mathsf{crs}, \mathsf{ek})$, we sample a simulated CRS $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{NIZK.Sim}(1^\lambda)$ instead of $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, and (2) every time we compute an NP NIZK proof π_P as part of the QMA NIZK proof $|\pi\rangle$, we use the NP NIZK simulator $\pi_\mathsf{P} \leftarrow \mathsf{NIZK.Sim}(\mathsf{td}, (\hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \mathsf{ek}))$ rather than $\pi_\mathsf{P} \leftarrow \mathsf{NIZK.P}(\mathsf{crs}, (\hat{\mathsf{ct}}_\mathsf{P}, \mathsf{ct}_{r_\Xi}, \mathsf{ek}))$ (where we execute $\mathsf{NIZK.P}$ along with a witness for the statement). To move to this hybrid we will use the adaptive zero knowledge property of the NP NIZK.

- Hyb_4 : This hybrid process is identical to Hyb_3, with the exception that ct_{r_Ξ} is an encryption of zeros rather than the randomness for the circuit C_{x,r_Ξ}, which is homomorphically evaluated. To move to this hybrid we will use the security of the PKE scheme.

- Hyb_5 : This hybrid process is identical to Hyb_4, with the exception that when computing the evaluated ciphertext $\hat{\mathsf{ct}}_\mathsf{P}$, instead of homomorphically evaluating the circuit C_{x,r_Ξ}, we compute C_{x,r_Ξ} in the clear and inject the result by circuit-private evaluation. More precisely, the prover does the following: First, it regularly computes $\alpha = \Xi.\mathsf{P}(|w\rangle^{\otimes k(\lambda)}; r_\Xi)$, for randomness r_Ξ. It derives prfk from the decrypted randomness r_V, computes $\beta_x = \mathsf{PRF.F}_{\mathsf{prfk}}(x)$, $\gamma = \Xi.\mathsf{P}_3(\beta_x, r_\Xi)$, and then $\hat{\mathsf{ct}}_\mathsf{P} \leftarrow \mathsf{FHE.Eval}(C_\gamma, \mathsf{ct}_\mathsf{V})$, where C_γ is the circuit that always outputs γ. To move to this hybrid we will use the circuit-privacy property of the FHE's evaluation algorithm.

- Hyb_6 : This hybrid process is identical to Hyb_5, with the exception that when computing (α, γ) we use the quantum sigma protocol (special zero-knowledge) simulator, that is, the prover first computes β_x (from prfk which is derived from r_V) and then computes $(\alpha, \gamma) \leftarrow \Xi.\mathsf{Sim}(x, \beta_x)$ and as before, α is sent in the clear and γ is sent through homomorphically evaluating the circuit C_γ on ct_V. To move to this hybrid we will use the special zero knowledge property of the quantum

sigma protocol. Note that the actions of the prover in this hybrid process are exactly the ones of the QMA NIZK simulator Sim and thus Hyb_6 is exactly the process described in the experiment of $P_{\mathsf{Simulated}}$.

We now claim that the outputs of each two consecutive hybrids are computationally indistinguishable, which will finish our proof.

- $\mathsf{Hyb}_0 \approx_s \mathsf{Hyb}_1$: Follows readily from the pseudorandomness property of the public keys generated by $\mathsf{PKE.Gen}(1^\lambda)$.
- $\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_2$: Follows from the adaptive soundness of the NIZK protocol for NP and the statistical correctness of the PKE scheme. We explain in more detail: First, note that whenever the NP statement that D^* proves in π_V is correct and the decryption of the PKE is correct, then the output distribution of the proof oracle is identical between the two hybrid processes, as the additional check that is made in Hyb_2 passes successfully. Also note that whenever the proof π_V is invalid, then both processes output $\perp$ and are identical. It follows that the only times that the output distributions of the proof oracles are not identical is whenever there is an error in the decryption of the PKE, or the proof π_V checks successfully but the statement is false i.e. whenever D^* breaks the adaptive soundness of the NP NIZK protocol. Since both of the above happen with at most negligible probability, it follows that only with negligible probability the outputs of Hyb_1 and Hyb_2 can be distinguished, and the statistical closeness between them follows.
- $\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_3$: Follows readily from the adaptive zero-knowledge property of the NP NIZK protocol.
- $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$: Follows from the security of the PKE scheme. Specifically, the encrypted randomness r_Ξ for every query is simply a random string (independent of all other operations in the process) and thus all of these random strings can be chosen at the beginning of the execution of the process, and thus we fix by an averaging argument the strings $r_\Xi^1, \cdots, r_\Xi^q$ that maximize the distinguishability of D^*, where the q is the (polynomial) number of queries that D^* makes to the proof oracle. It then follows that if D^* distinguishes between Hyb_3 and Hyb_4 then it distinguishes between encryptions of $r_\Xi^1, \cdots, r_\Xi^q$ and encryptions of zeros, and since the single-message security of public-key encryption schemes implies many-message security the indistinguishability $\mathsf{Hyb}_3 \approx_c \mathsf{Hyb}_4$ follow.
- $\mathsf{Hyb}_4 \approx_s \mathsf{Hyb}_5$: Follows by a hybrid argument, by the circuit-privacy property of the FHE scheme and from the fact that the prover makes the additional check on the public verification key, which checks that ct_V is obtained by running $\mathsf{PRF.Gen}, \mathsf{fhek} \leftarrow \mathsf{FHE.Gen}, \mathsf{FHE.Enc}_{\mathsf{fhek}}$ with the extracted randomness r_V. More precisely, let q be the number of queries that D^* makes to the proof oracle, and for $i \in \{0, 1, \cdots, q\}$ we define Hyb_4^i as the process that performs the homomorphic evaluation of C_{x, r_Ξ} (rather than computing it in the clear and then injecting the result, as done in Hyb_5) starting from query number $i+1$ that D^* makes, thus $\mathsf{Hyb}_4^0 = \mathsf{Hyb}_4$, $\mathsf{Hyb}_4^q = \mathsf{Hyb}_5$.
  If Hyb_4 and Hyb_5 are distinguishable then for some $i \in \{0, 1, \cdots, q-1\}$, Hyb_4^i and Hyb_4^{i+1} are distinguishable. We fix by an averaging argument a snapshot of the execution until after the point that D^* sends the $(i+1)$-th query to the proof oracle.

27

If the check that the prover makes in the beginning, which includes both checking the validity of the NP proof π_V and also checking the validity of creating ct_V from the extracted randomness r_V, fails, then the hybrid processes are the same as the answer of the proof oracle will be $\bot$. In case the check is successful, it follows that the outputs of the circuits C_{x,r_Ξ} and C_γ on the input prfk (which is encrypted inside ct_V) are the same, and thus it follows that the distinguisher between the hybrids Hyb_4^i and Hyb_4^{i+1} can be used to break the (even statistical) circuit privacy of the FHE evaluation.

- $\mathsf{Hyb}_5 \approx_c \mathsf{Hyb}_6$: The proof is very similar to the proof for the indistinguishability $\mathsf{Hyb}_4 \approx_c \mathsf{Hyb}_5$, as the indistinguishability follows by a hybrid argument and from the special zero knowledge property of the quantum sigma protocol. More precisely, for $i \in \{0, 1, \cdots, q\}$ we define Hyb_5^i as the process that uses $\Xi.\mathsf{P}$ (and the polynomially-many copies of the quantum witness) in order to generate (α, γ) (rather than computing it using the simulator) starting from query number $i+1$ that D^* makes, thus $\mathsf{Hyb}_5^0 = \mathsf{Hyb}_5$, $\mathsf{Hyb}_5^q = \mathsf{Hyb}_6$.

  If Hyb_5 and Hyb_6 are distinguishable then for some $i \in \{0, 1, \cdots, q-1\}$, Hyb_5^i and Hyb_5^{i+1} are distinguishable. We fix by an averaging argument a snapshot of the execution until after the point that D^* sends the $(i+1)$-th query to the proof oracle, this in particular fixes the yes instance $x \in \mathcal{L}_{yes}$, the quantum witness $|w\rangle$ and the pseudorandomness β_x. It follows that the distinguisher between the hybrids Hyb_5^i and Hyb_5^{i+1} can be used to tell the difference between a tuple (α, γ) that was generated by $\Xi.\mathsf{P}$ and a tuple that was generated by $\Xi.\mathsf{Sim}$, in contradiction the special zero knowledge property of the protocol $(\Xi.\mathsf{P}, \Xi.\mathsf{V})$.

References

ACGH19. Gorjan Alagic, Andrew M Childs, Alex B Grilo, and Shih-Han Hung. Non-interactive classical verification of quantum computation. *arXiv*, pages arXiv–1911, 2019.

BCKM20. James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. On the round complexity of two-party quantum computation. *arXiv preprint arXiv:2011.11212*, 2020.

BD18. Zvika Brakerski and Nico Döttling. Two-message statistically sender-private ot from lwe. In *Theory of Cryptography Conference*, pages 370–390. Springer, 2018.

BFM19. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 329–349. 2019.

BG19. Anne Broadbent and Alex B Grilo. Zero-knowledge for qma from locally simulatable proofs. *arXiv preprint arXiv:1911.07782*, 2019.

BJSW16. Anne Broadbent, Zhengfeng Ji, Fang Song, and John Watrous. Zero-knowledge proof systems for qma. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 31–40. IEEE, 2016.

BKM20. Zvika Brakerski, Venkata Koppula, and Tamer Mour. Nizk from lpn and trapdoor hash via correlation intractability for approximable relations. *IACR Cryptol. ePrint Arch.*, 2020:258, 2020.

Blu86. Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, volume 1, page 2. Citeseer, 1986.

BP15.   Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *Theory of Cryptography Conference*, pages 401–427. Springer, 2015.

CCH+19.   Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N Rothblum, Ron D Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1082–1090, 2019.

CGH04.   Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.

CVZ19.   Andrea Coladangelo, Thomas Vidick, and Tina Zhang. Non-interactive zero-knowledge arguments for qma, with preprocessing. *arXiv preprint arXiv:1911.07546*, 2019.

FLS99.   Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on computing*, 29(1):1–28, 1999.

GMR89.   Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

KW19.   Sam Kim and David J Wu. Multi-theorem preprocessing nizks from lattices. *Journal of Cryptology*, pages 1–84, 2019.

Liu06.   Yi-Kai Liu. Consistency of local density matrices is qma-complete. In *Approximation, randomization, and combinatorial optimization. algorithms and techniques*, pages 438–449. Springer, 2006.

LQR+19.   Alex Lombardi, Willy Quach, Ron D Rothblum, Daniel Wichs, and David J Wu. New constructions of reusable designated-verifier nizks. In *Annual International Cryptology Conference*, pages 670–700. Springer, 2019.

Mor20.   Tomoyuki Morimae. Information-theoretically-sound non-interactive classical verification of quantum computing with trusted center. *arXiv preprint arXiv:2003.10712*, 2020.

MY21.   Tomoyuki Morimae and Takashi Yamakawa. Classically verifiable (dual-mode) nizk for qma with preprocessing. *arXiv preprint arXiv:2102.09149*, 2021.

OPCPC14. Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private fhe. In *Annual Cryptology Conference*, pages 536–553. Springer, 2014.

PS19.   Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for np from (plain) learning with errors. In *Annual International Cryptology Conference*, pages 89–114. Springer, 2019.

PV+06.   Rafael Pass, Vinod Vaikuntanathan, et al. Construction of a non-malleable encryption scheme from any semantically secure one. In *Annual International Cryptology Conference*, pages 271–289. Springer, 2006.

QRW19.   Willy Quach, Ron D Rothblum, and Daniel Wichs. Reusable designated-verifier nizks for all np from cdh. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 593–621. Springer, 2019.

Reg09.   Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.