Electronic Colloquium on Computational Complexity, Report No. 86 (2019)

Perfect zero knowledge for quantum multiprover interactive proofs

Alex B. Grilo [*] William Slofstra [†] Henry Yuen [‡]

Abstract

In this work we consider the interplay between multiprover interactive proofs, quantum entanglement, and zero knowledge proofs — notions that are central pillars of complexity theory, quantum information and cryptography. In particular, we study the relationship between the complexity class MIP^*, the set of languages decidable by multiprover interactive proofs with quantumly entangled provers, and the class $\mathsf{PZK\text{-}MIP}^*$, which is the set of languages decidable by MIP^* protocols that furthermore possess the *perfect zero knowledge* property.

Our main result is that the two classes are equal, i.e., $\mathsf{MIP}^* = \mathsf{PZK\text{-}MIP}^*$. This result provides a quantum analogue of the celebrated result of Ben-Or, Goldwasser, Kilian, and Wigderson (STOC 1988) who show that $\mathsf{MIP} = \mathsf{PZK\text{-}MIP}$ (in other words, all classical multiprover interactive protocols can be made zero knowledge). We prove our result by showing that every MIP^* protocol can be efficiently transformed into an equivalent zero knowledge MIP^* protocol in a manner that preserves the completeness-soundness gap. Combining our transformation with previous results by Slofstra (Forum of Mathematics, Pi 2019) and Fitzsimons, Ji, Vidick and Yuen (STOC 2019), we obtain the corollary that all co-recursively enumerable languages (which include undecidable problems as well as all decidable problems) have zero knowledge MIP^* protocols with vanishing promise gap.

[*]CWI and QuSoft, Amsterdam, The Netherlands. alexg@cwi.nl
[†]IQC and Department of Pure Mathematics, University of Waterloo, Waterloo, Canada. weslofst@uwaterloo.ca
[‡]University of Toronto, Toronto, Canada. hyuen@cs.toronto.edu

ISSN 1433-8092

1 Introduction

Multiprover interactive proofs (MIPs) are a model of computation where a probabilistic polynomial time verifier interacts with several all-powerful — but non-communicating — provers to check the validity of a statement (for example, whether a quantified boolean formula is satisfiable). If the statement is true, then there is a strategy for the provers to convince the verifier of this fact. Otherwise, for all prover strategies, the verifier rejects with high probability. This gives rise to the complexity class MIP, which is the set of all languages that can be decided by MIPs. This model of computation was first introduced by Ben-Or, Goldwasser, Kilian and Wigderson [6]. A foundational result in complexity theory due to Babai, Fortnow, and Lund shows that multiprover interactive proofs are surprisingly powerful: MIP is actually equal to the class of problems solvable in non-deterministic exponential time, i.e., $\mathsf{MIP} = \mathsf{NEXP}$ [2].

Research in quantum complexity theory has led to the study of *quantum* MIPs. In one of the most commonly considered models, the verifier interacts with provers that are *quantumly entangled*. Even though the provers still cannot communicate with each other, they can utilize correlations arising from local measurements on entangled quantum states. Such correlations cannot be explained classically, and the study of the counter-intuitive nature of these correlations dates back to the famous 1935 paper of Einstein, Podolsky and Rosen [18] and the seminal work of Bell in 1964 [4]. Over the past twenty years, MIPs with entangled provers have provided a fruitful computational lens through which the power of such correlations can be studied. The set of languages decidable by such interactive proofs is denoted by MIP^*, where the asterisk denotes the use of entanglement.

Finally, another type of interactive proof system are zero knowledge proofs. These were introduced by Goldwasser, Micali and Rackoff [21] and have played a crucial role in the development of theoretical cryptography. In this model, if the claimed statement is indeed true, the interaction between the verifier and prover must be conducted in such a way that the verifier learns *nothing else aside from the validity of the statement*. This is formalized by requiring the existence of an efficient *simulator* whose output is indistinguishable from the distribution of the messages in a real execution of the protocol. It was shown by [6] that any (classical) MIP protocol can be transformed into an equivalent perfect zero knowledge[1] MIP protocol. In other words, the complexity classes MIP (and thus NEXP) and PZK-MIP are equal, where the latter consists of all languages decidable by perfect zero knowledge MIPs.

Informally stated, our main result is a quantum analogue of the result of Ben-Or, Goldwasser, Kilian, and Wigderson [6]: we show that

Every MIP protocol can be efficiently transformed into an equivalent zero knowledge MIP* protocol.*

Phrased in complexity-theoretic terms, we show that $\mathsf{MIP}^* = \mathsf{PZK\text{-}MIP}^*$. This is a strengthening of the recent results of Chiesa, Forbes, Gur and Spooner, who show that $\mathsf{NEXP} = \mathsf{MIP} \subseteq \mathsf{PZK\text{-}MIP}^*$ [10] (which is, in turn, a strengthening of the the result of Ito and Vidick that $\mathsf{NEXP} \subseteq \mathsf{MIP}^*$ [24]).

Surprisingly, there are no upper bounds known on the power of quantum MIPs. The recent spectacular result of Natarajan and Wright shows that MIP^* contains the complexity class NEEXP, which is the enormously powerful class of problems that can be solved in non-deterministic *doubly exponential* time [32]. Since $\mathsf{NEXP} \neq \mathsf{NEEXP}$ via the non-deterministic time hierarchy theorem [14], this unconditionally shows that quantum MIPs are strictly more powerful than classical MIPs. Furthermore, it is conceivable that MIP^* even contains *undecidable* languages. In [36, 37], Slofstra proved that determining whether a given MIP^* protocol admits a prover strategy that wins with certainty is an undecidable problem. In [19], Fitzsimons, Ji, Vidick and Yuen showed that the

[1]The term *perfect* refers to the property that the interaction in a real protocol can be simulated without any error.

class $\mathsf{MIP}^*_{1,1-\varepsilon(n)}$, the set of languages decidable by MIPs protocols with *promise gap* $\varepsilon(n)$ that can depend on the input size, contains $\mathsf{NTIME}[2^{\mathrm{poly}(1/\varepsilon(n))}]$, the class of problems that are solvable in non-deterministic time $2^{\mathrm{poly}(1/\varepsilon(n))}$. In contrast, the complexity of MIP (even with a shrinking promise gap) is always equal to NEXP.

Thus, our result implies that all languages in NEEXP – and any larger complexity classes discovered to be contained within MIP^* – have perfect zero knowledge interactive proofs with entangled provers. In fact, we prove a stronger statement: every MIP^* protocol with promise gap ε also has an equivalent zero knowledge MIP^* protocol with promise gap that is polynomially related to ε. This, combined with the results of [19] and [37], implies that languages of arbitrarily large time complexity – including some undecidable problems – have zero knowledge proofs (albeit with vanishing promise gap).

1.1 Our results

We state our results in more detail. Let $\mathsf{MIP}^*_{c,s}[k,r]$ denote the set of languages L that admit k-prover, r-round MIP^* protocols with completeness c, and soundness s. In other words, there exists a probabilistic polynomial-time verifier V that interacts with k entangled provers over r rounds so that if $x \in L$, then there exists a prover strategy that causes $V(x)$ to accept with probability at least c; [2] otherwise all prover strategies cause $V(x)$ to accept with probability strictly less than s. The class $\mathsf{PZK\text{-}MIP}^*_{c,s}[k,r]$ are the languages that have $\mathsf{MIP}^*[k,r]$ protocols where the interaction between the verifier can be simulated exactly and efficiently, without the aid of any provers. We provide formal definitions of these complexity classes in Section 2.3.

In what follows, let n denote the input size. The parameters k, r, s of a protocol are also allowed to depend on the input size. In this paper, unless stated otherwise, we assume that completeness parameter c in a protocol is equal to 1.

Theorem 1. *For all $0 \le s \le 1$, for all polynomially bounded functions k, r,*

$$\mathsf{MIP}^*_{1,s}[k,r] \subseteq \mathsf{PZK\text{-}MIP}^*_{1,s'}[k+4,1]$$

where $s' = 1-(1-s)^\alpha$ for some universal constant $\alpha > 0$.

The first corollary of Theorem 1 concerns what we call *fully quantum* MIPs, which are multiprover interactive proofs where the verifier can perform polynomial time quantum computations and exchange quantum messages with entangled quantum provers. The set of languages decidable by fully quantum MIPs is denoted by QMIP, which clearly includes MIP^*. Reichardt, Unger, and Vazirani [35] showed that the reverse inclusion also holds by adding two additional provers; i.e., that $\mathsf{QMIP}[k] \subseteq \mathsf{MIP}^*[k+2]$. Combined with Theorem 1 and the fact that we can assume that QMIP protocols have perfect completeness if we add an additional prover (see [38]), this implies that

Corollary 2. *For all polynomially bounded functions k, r, we have*

$$\mathsf{QMIP}_{1,\frac{1}{2}}[k,r] \subseteq \mathsf{PZK\text{-}MIP}^*_{1,\frac{1}{2}}[k+4,1].$$

The combination of the results in [19] and [32] implies that for every hyper-exponential function f,[3] we have that

$$\mathsf{NTIME}[2^{2^{f(n)}}] \subseteq \mathsf{MIP}^*_{1,s}[4,1],$$

[2]Technically speaking, the completeness condition actually corresponds to a *sequence* of prover strategies with success probability approaching c; we discuss this subtlety in Section 2.4.

[3]A hyper-exponential function $f(n)$ is of the form $\exp(\cdots\exp(\mathrm{poly}(n))\cdots)$, where the number of iterated exponentials is $R(n)$ for some time-constructible function $R(n)$.

where $\mathsf{NTIME}[g(n)]$ denotes the set of languages that can be decided by nondeterministic Turing machines running in time $g(n)$ and $s = 1 - Cf(n)^{-c}$ for some universal constants C and c, independent of n.[4] Combining this with Theorem 1, we obtain the following.

Corollary 3. *There exist universal constants $C, c > 0$ such that for all hyper-exponential functions $f : \mathbb{N} \to \mathbb{N}$,*

$$\mathsf{NTIME}[2^{2^{f(n)}}] \subseteq \mathsf{PZK\text{-}MIP}^*_{1,s}[6,1]$$

where $s = 1 - Cf(n)^{-c}$.

Finally, it was also shown in [19, 37] that the undecidable language NONHALT, which consists of Turing machines that do not halt when run on the empty input tape, is contained in $\mathsf{MIP}^*_{1,1}[2,1]$. The "$1,1$" subscript indicates that for negative instances (i.e., Turing machines that do halt), the verifier rejects with positive probability. In more detail: there exists a polynomial time computable function that maps Turing machines M to an MIP^* protocol V_M such that if M does not halt on the empty input tape, then there is a prover strategy for V_M that is accepted with probability 1; otherwise there exists a positive constant $\varepsilon > 0$ (depending on M) such that for all prover strategies, the protocol V_M rejects with probability ε.

Theorem 1 implies there is a polynomial time computable mapping $V_M \mapsto V'_M$ such that V'_M is a $\mathsf{PZK\text{-}MIP}^*$ protocol that preserves completeness (if V_M accepts with probability 1, then so does V'_M) and soundness (if V_M rejects with probability ε for all prover strategies, then V'_M rejects with probability $\mathrm{poly}(\varepsilon)$ for all prover strategies). Therefore, we can conclude the following:

Corollary 4. $\mathrm{NONHALT} \in \mathsf{PZK\text{-}MIP}^*_{1,1}[4,1]$.

Corollary 4 implies that all co-recursively enumerable languages (languages whose complement are recursively enumerable) have zero knowledge proofs (with vanishing gap).

1.2 Proof overview

The proof of Theorem 1 draws upon a number of ideas and techniques that have been developed to study interactive protocols with entangled provers. At a high level, the proof proceeds as follows. Let L be a language that is decided by some k-prover MIP^* protocol with a verifier V. Assume for simplicity that on positive instances $x \in L$, there is a prover strategy that causes V to accept with probability 1, and otherwise rejects with high probability. Although V is probabilistic polynomial time (PPT) Turing machine in a MIP^* protocol, we can instead think of it as a quantum circuit involving a combination of verifier computations, and prover computations.

First, we transform the verifier V into an equivalent quantum circuit V_{enc} where the computation is now performed on *encoded data*. We do this using techniques from quantum fault-tolerance, where the data is protected using a quantum error correcting code, and physical operations are performed on the encoded data in order to effect logical operations on the underlying logical data.

We then apply *protocol compression* to V_{enc} to obtain a new verifier V_{ZK} for an equivalent protocol — this will be our zero knowledge MIP^* protocol. Protocol compression is a technique that was pioneered by Ji in [25] (and further developed by Fitzsimons, Ji, Vidick and Yuen [19]) to show that NEXP has 1-round MIP^* protocols where the communication is logarithmic length. Essentially, in

[4]The original result in [19] states that for all hyper-exponential functions $f(n)$, $\mathsf{NTIME}[2^{f(n)}] \subseteq \mathsf{MIP}^*_{1,s}[15,1]$ for $s = 1 = Cf(n)^{-c}$. Using a more efficient error correcting code as described in Section A, the number of provers can be reduced to 4. The improvement from $\mathsf{NTIME}[2^{f(n)}]$ to $\mathsf{NTIME}[2^{2^{f(n)}}]$ is obtained by plugging in the $\mathsf{NEEXP} \subseteq \mathsf{MIP}^*$ result of Natarajan and Wright [32] as the "base case" of the iterated compression scheme, instead of the $\mathsf{NEXP} \subseteq \mathsf{MIP}^*$ result of Natarajan and Vidick [31].

the compressed protocol, the new verifier V_{ZK} efficiently checks whether V_{enc} would have accepted in the original protocol without actually having to run V_{enc}, by testing that the provers hold an entangled *history state* of a successful interaction between V_{enc} and some provers.

The reason this compressed protocol is zero knowledge is the following: the verifier V_{ZK} asks the provers to report the outcomes of performing local measurements in order to verify that they hold an accepting history state. In the positive case (i.e., $x \in L$), there is an "honest" strategy where the provers share a history state $|\Phi\rangle$ of a successful interaction with V_{enc}. We argue that, because of the fault-tolerance properties of V_{enc}, individual local measurements on $|\Phi\rangle$ reveal no information about the details of the interaction. Put another way, the distribution of outcomes of honest provers' local measurements can be efficiently simulated, without the aid of any provers at all. Since we only require that this simulatability property holds with respect to honest provers, this establishes the zero knowledge property of the protocol run by V_{ZK}.

In the next few sections, we provide more details on the components of this transformation. We discuss things in reverse order: first, we give an overview of the protocol compression technique. Then, we discuss the fault tolerant encoding V_{enc} of the original verifier V. Then we describe how applying protocol compression to V_{enc} yields a zero knowledge protocol for L.

1.2.1 Protocol compression

The protocol compression technique of [25, 19] transforms any k-prover, r-round QMIP protocol where the verifier V runs in time N into a $k+O(1)$-prover, 1-round MIP* protocol where the verifier V' runs in time $\operatorname{poly}\log N$. In other words, the verifier has been compressed into an exponentially more efficient one; however, this comes with the price of having the promise gap shrink as well: if the promise gap of the original QMIP protocol is ε, then the promise gap of the compressed protocol is $\operatorname{poly}(\varepsilon/N)$.

This compression is achieved as follows: in the protocol executed by the compressed verifier V', the provers are tested to show that they possess an (encoding of) a *history state* of the original protocol executed by V, describing an execution of the protocol in which the original verifier V accepts. History states of some T-length computation generally look like the following:

$$|\psi\rangle = \frac{1}{\sqrt{T+1}} \sum_{t=0}^{T} |t\rangle \otimes |\psi_t\rangle.$$

The first register holding the superposition over $|t\rangle$ is called the *clock register*; the second register holding the superposition over $|\psi_t\rangle$ is called the *snapshot register*. The t-th snapshot of the computation $|\psi_t\rangle$ is the global state of the protocol at time t:

$$|\psi_t\rangle = g_t g_{t-1} \cdots g_1 |\psi_0\rangle$$

where the g_i's are the gates used in the protocol, and $|\psi_0\rangle$ is the initial state of the protocol. Usually, each g_i is a one- or two-qubit gate that is part of the verifier V's computation. However, g_i could also represent a *prover gate*, which is the computation performed by one of the k provers. Unlike gates in the verifier's computation, the prover gates are non-local, and there is no characterization of their structure. In general, they may have exponential circuit complexity, and may act on a Hilbert space that can be much larger than the space used by the verifier V.

This notion of history states for interactive protocols is a generalization of the basic concept of history states for quantum circuits, which was introduced by Kitaev to prove that the local Hamiltonians problem is QMA-complete [27]. He showed that for every QMA verifier circuit C, there exists a local Hamiltonian $H(C)$ (called the Feynman-Kitaev Hamiltonian) such that all ground states of $H(C)$ are history states of the circuit C. To test whether a given state $|\phi\rangle$ is a history state of C, one can sample random terms from $H(C)$ and measure them to get an estimate of the energy of $|\phi\rangle$ with respect to $H(C)$.

In slightly more detail, the local Hamiltonian $H(C)$ consists of terms that can be divided into four groups:

- *Input checking terms* H_{in}. These terms check that the initial snapshot $|\psi_0\rangle$, which represents the initial state of the QMA verifier, has all of its ancilla bits set to zero.
- *Clock checking terms* H_{clock}. These terms check that the clock register is encoded in unary. The unary encoding is to ensure that the locality of $H(C)$ is a fixed constant independent of the computation.
- *Propagation terms* H_{prop}. These terms check that the history state is a uniform superposition over snapshots $|\psi_t\rangle$, with $|\psi_t\rangle = g_t|\psi_{t-1}\rangle$.
- *Output checking terms* H_{out}. These terms check that at time $t = T$, the decision bit of the QMA verifier is equal to $|1\rangle$ (i.e., the verifier accepted).

In [25], Ji showed that for every quantum interactive protocol Π, there is a *generalized protocol Hamiltonian* $H(\Pi)$ whose ground states are all history states of Π. The Hamiltonian $H(\Pi)$ is essentially the Feynman-Kitaev Hamiltonian corresponding to the verifier V, except if at time t in the protocol Π, prover i is supposed to implement a unitary g_t on their registers (which includes their private registers as well as some registers used to communicate with the verifier), then there will be a corresponding non-local propagation term

$$\frac{1}{2}\left(|t-1\rangle \otimes I - |t\rangle \otimes g_t\right)\left(\langle t-1| \otimes I - \langle t| \otimes g_t^{\dagger}\right). \tag{1}$$

This term is non-local because of the prover gate g_t, which may act on a Hilbert space of unbounded size. Other than these prover propagation terms, the rest of $H(\Pi)$ corresponds to the local computations performed by the verifier V.

Suppose that one had the ability to sample random terms of $H(\Pi)$ and efficiently measure a given state with the terms. Then, by performing an energy test on a state $|\psi\rangle$, one could efficiently determine whether the state was close to a history state that describes an accepting interaction in the protocol Π. This appears to be a difficult task for terms like (1) when g_t is a prover gate, since this requires performing a complex non-local measurement. Furthermore, the tester would not know what prover strategy to use.

Ji's insight in [25] was that a tester could efficiently *delegate* the energy measurements to entangled quantum provers. He constructs a protocol where the verifier V' commands the provers to perform measurements corresponding to random terms of $H(\Pi)$ on their shared state. If the reported energy is low, then V' is convinced that there must exist a history state of Π that describes an accepting interaction (and in particular, the provers share this history state).

In order to successfully command the provers, the verifier V' relies on a phenomena called *non-local game rigidity* (also known as *self-testing*). Non-local games are one-round protocols between a classical verifier and multiple entangled provers. This phenomena is best explained using the famous CHSH game, which is a two-player game where the optimal entangled strategy succeeds with probability $\omega^*(CHSH) = \frac{1}{2} + \frac{1}{\sqrt{2}}$. The canonical, textbook strategy for CHSH is simple: the two players share a maximally entangled pair of qubits, and measure their respective qubits using the Pauli observables σ_X and σ_Z, depending on their input. The rigidity property of the CHSH game implies that this canonical strategy is, in some sense, *unique*: *any* optimal entangled strategy for CHSH must be, up to a local basis change, identical to this canonical strategy. Thus we also say that the CHSH game is a *self-test* for a maximally entangled pair of qubits and single-qubit Pauli measurements for the players.

There has been extensive research on rigidity of non-local games [35, 15, 28, 12, 9, 29, 30, 13], and many different self-tests have been developed. The non-local games used in the compression protocols of [25, 19] are variants of the CHSH game, where the canonical optimal strategy is roughly the following: the players share a maximally entangled state on n qubits, and their measurements are tensor products of Pauli observables on a constant number of those n qubits, such as

$$\sigma_X(i) \otimes \sigma_Z(j) \otimes \sigma_Z(k).$$

which indicates σ_X acting on the i'th qubit, and σ_Z on the j'th and k'th. This game also has the following robust self-testing guarantee: any entangled strategy that succeeds with probability $1-\varepsilon$ must be $\mathrm{poly}(\varepsilon, n)$-close to the canonical strategy. Here, n is a growing parameter, whereas the weight of the Pauli observables (i.e. the number of factors that don't act as the identity) is at most some constant independent of n.

For the terms of $H(\Pi)$ that involve uncharacterized prover gates, the verifier V' simply asks some provers to measure the observable corresponding to the prover gate. By carefully interleaving rigidity tests with the energy tests, the verifier V' can ensure that the provers are performing the desired measurements for all other terms of $H(\Pi)$, and thus test if they have an accepting history state.

1.2.2 Quantum error correction and fault tolerant verifiers

In order to describe our fault tolerant encoding of verifiers, we first discuss quantum error correction and fault tolerant quantum computation.

Quantum error correcting codes (QECCs) provide a way of encoding quantum information in a form that is resilient to noise. Specifically, a $[[n, k, d]]$ quantum code $\mathcal{C}$ encodes all k-qubit states $|\psi\rangle$ into an n-qubit state $\mathsf{Enc}(|\psi\rangle)$ such that for any quantum operation $\mathcal{E}$ that acts on at most $(d-1)/2$ qubits, the original state $|\psi\rangle$ can be recovered from $\mathcal{E}(\mathsf{Enc}(|\psi\rangle))$. The parameter d is known as the *distance* of the code $\mathcal{C}$.

QECCs are an important component of *fault tolerant* quantum computation, which is a method for performing quantum computations in a way that is resilient to noise. In a fault tolerant quantum computation, the information $|\psi\rangle$ of a quantum computer is encoded into a state $\mathsf{Enc}(|\psi\rangle)$ using some QECC $\mathcal{C}$, and the computation operations are performed on the encoded data without ever fully decoding the state.

For example, in many stabilizer QECCs, in order to compute $\mathsf{Enc}(g|\psi\rangle)$ for some single-qubit Clifford gate g, it suffices to apply g *transversally*, i.e., apply g on every physical qubit of $\mathsf{Enc}(|\psi\rangle)$. Transversal operations are highly desirable in fault tolerant quantum computation because they spread errors in a controlled fashion.

Non-Clifford gates, however, do not admit a transversal encoding in most stabilizer QECCs. In order to implement logical non-Clifford gates, one can use *magic states*. These are states that encode the behaviour of some non-Clifford gate g (such as a Toffoli gate, or a $\pi/8$ rotation), and are prepared and encoded before the computation begins. During the fault tolerant computation, the encoded magic states are used in *gadgets* that effectively apply the non-Clifford g to the encoded data. These gadgets only require measurements and transversal Clifford operations that are controlled on the classical measurement outcomes.

We now discuss the behaviour of the verifier V_{enc}. First, the encoded verifier spends time manufacturing a collection of encoded ancilla states, as well as encoded magic states of some non-Clifford gates (in our case, the Toffoli gate), using some fixed quantum error correcting code $\mathcal{C}$. We call this the **Resource Generation Phase**. Then, the verifier V_{enc} simulates the execution of V on the encoded information from the Resource Generation Phase. All Clifford operations of V are performed transversally, and non-Clifford operations of V are performed with the help of the encoded magic states. When interacting with the provers, the verifier V_{enc} sends its messages in encoded form as well – the provers are capable of decoding and re-encoding messages using the code $\mathcal{C}$.

Finally, after the finishing the simulation of V, the verifier V_{enc} executes an **Output Decoding Phase**: it performs a decoding procedure on the physical qubits corresponding to the output qubit of V.

It is clear that the protocol executed by V_{enc} is equivalent to the protocol executed by V. The overhead introduced by this fault tolerant encoding is a constant factor increase in the length of the circuit (depending on the size of the code $\mathcal{C}$). The fault tolerant properties of the computation of V_{enc} will play a major role in our proof of zero knowledge.

1.2.3 The zero knowledge protocol, and its analysis

To distinguish between the parties of the "inner" protocol executed by V_{enc} and the parties in the "outer" protocol executed by V_{ZK}, we say that V_{enc} is a *verifier* that interacts with a number of *provers*. On the other hand, we say that V_{ZK} is a *referee* that interacts with a number of *players*.

The zero knowledge protocol executed by V_{ZK} consists of applying protocol compression to the fault tolerant verifier V_{enc}. The result is a MIP* protocol that checks whether the players possess a history state of an accepting interaction with V_{enc} and some provers.

The formal definition of the zero knowledge property requires an efficient algorithm, called the simulator, that when given a yes instance (i.e., $x \in L$), produces an output that is identically distributed to the transcript produced by an interaction between the referee and players following a specified *honest strategy*. The interaction must be simulatable even when the referee doesn't follow the protocol. A cheating referee could, for instance, sample questions differently than the honest referee, or interact with the players in a different order. The only constraint we have is that the *format* of the questions, from the perspective of an individual player, must look like something the honest referee *could have* sent. In particular, if a cheating referee tries to interact with an individual player multiple times, the player would abort the protocol.

In the yes instance, the honest player strategy for V_{ZK} consists of sharing a history state $|\Phi\rangle$ that describes the referee V_{enc} interacting with some provers and accepting with probability 1. When the players receive a question in V_{ZK}, they either measure some Pauli observable on a constant number of qubits of $|\Phi\rangle$, or measure the observable corresponding to a prover gate. The zero knowledge property of V_{ZK} rests on the ability to efficiently sample the outcomes of measurements formed from *any* combination of local Pauli observables and prover measurements that might be commanded by a cheating referee.

We first analyze *non-adaptive* referees; that is, they sample the questions to all the players first. In the compressed protocol V_{ZK}, the honest referee asks the players to perform local measurements corresponding to a random term in the the Hamiltonian $H(\Pi)$. Thus, the support of the measurements commanded by a referee (even a cheating one) can only involve a constant number of qubits of $|\Phi\rangle$. Let $\widehat{W}$ denote the tuple of questions sent to the players, and let $S_{\widehat{W}}$ denote the registers of $|\Phi\rangle$ that are supposed to be measured. We argue that the reduced density matrix $|\Phi\rangle$ on the registers $S_{\widehat{W}}$ can be computed explicitly in polynomial time.

This is where the fault tolerance properties of V_{enc} come in. Since V_{enc} is running a computation on encoded information, any local view of the state of V_{enc} in the middle of its computation should not reveal any details about the actual information being processed. Intuitively, the purpose of a quantum error correcting code is to conceal information from an environment that is making local measurements. In the zero knowledge context, we can think of the cheating referee as the "noisy environment" to V_{enc}. Thus, the cheating referee should not be able to learn anything because it can only access local correlations, while all the "juicy" information about V_{enc} is encoded in global correlations of $|\Phi\rangle$.

Although this is the high level idea behind our proof, there are several challenges that need to be overcome in order to make this argument work. First, the state of V_{enc} is not always properly encoded in an error correcting code: it may be in the middle of some logical operations, so there is a risk that some information may be leaked. We argue that if the code used by V_{enc} is *simulatable* (see Section 2.2), then this cannot happen. We show that the concatenated Steane code is simulatable, by analyzing coherent implementations of logical operations that do not reveal any information.

The next challenge is that the referee is able to perform local measurements not only on intermediate states of V_{enc} during its computation, but also *superpositions* of them. This threatens to circumvent the concealing properties of the error correcting code, because of the following example: suppose that $|\psi_0\rangle$ and $|\psi_1\rangle$ are orthogonal n qubit states such that the reduced density matrix of every small-sized subset of qubits of $|\psi_0\rangle$ or $|\psi_1\rangle$ looks maximally mixed. However, $\frac{1}{\sqrt{2}}\left(|0\rangle|\psi_0\rangle + |1\rangle|\psi_1\rangle\right)$ can be distinguished from $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi_0\rangle$ via a local measurement (namely, an σ_X measurement on the first qubit). One potential worry is that $|\psi_0\rangle$ and $|\psi_1\rangle$ might represent snapshots of the history state $|\Phi\rangle$ that are separated by many time steps, and therefore a simulator would have trouble simulating measurements on these superpositions, because it will not be able to determine what the inner product between $|\psi_0\rangle$ and $|\psi_1\rangle$ is in general.

We argue that, because of the structure of the protocol and the honest strategy, the cheating referee can only measure a superpositions that involve only constantly many consecutive snapshots of V_{enc}. From this we deduce that reduced density matrices of the superpositions can be efficiently computed.

Another challenge involves simulating the outcomes of measuring the prover gate, which may perform some arbitrarily complex computation. We carefully design the honest strategy for the compressed protocol so that measurement outcomes of the prover gate are always either constant, or an unbiased coin flip.

Finally, we argue that we can efficiently simulate the interaction of the protocol even when the referee behaves adaptively. The simulator for the non-adaptive case actually computes the reduced density matrix of the honest players' state; we can perform post-selection on the density matrix at most a polynomial number of times in order to simulate the distribution of questions and answers between an adaptive referee and the provers.

1.3 Related work

In this section, we discuss some relevant work on quantum analogues of zero knowledge proofs.

In quantum information theory, zero knowledge proofs have been primarily studied in the context of *single prover* quantum interactive proofs. This setting was first formalized by Watrous [39], and has been an active area of research over the years. Various aspects of zero knowledge quantum interactive proofs have been studied, including honest verifier models [39, 8], computational zero knowledge proof systems for QMA [7], and more.

In the multiprover setting, Chiesa, Forbes, Gur and Spooner [10] showed that all problems in

NEXP (and thus MIP) are in PZK-MIP$^*[2, \mathrm{poly}(n)]$. Their approach is considerably different of ours. They achieve their result by showing that model of interactive proofs called *algebraic interactive PCPs* [5] can be lifted to the entangled provers setting in a way that preserves zero knowledge, and then showing that languages in NEXP have zero knowledge algebraic interactive PCPs.

The results of [10] are, strictly speaking, incomparable to ours. We show that all languages in MIP* have single-round PZK-MIP* protocols with four additional provers, whereas [10] show that MIP (which is a subset of MIP*) have PZK-MIP* protocols with *two* provers and polynomially many rounds. Improving our result to only two provers seems to be quite a daunting challenge, as it is not even known how MIP$^*[k]$ relates to MIP$^*[k+1]$ – it could potentially be the case that adding more entangled provers yields a strictly larger complexity class!

Furthermore, the proof techniques of [10] are very different from ours: they heavily rely on algebraic PCP techniques, as well as the analysis of the low degree test against entangled provers [31]. Our proof relies on techniques from fault tolerant quantum computing and the protocol compression procedure of [25, 19], which in turn rely heavily on self-testing and history state Hamiltonians.

Another qualitative difference between the zero knowledge protocol of [10] and ours is that the honest prover strategy for their protocol does not require any entanglement; the provers can behave classically. In our protocol, however, the provers are required to use entanglement; this is what enables the class MIP* and PZK-MIP* to contain classes beyond NEXP, such as NEEXP (and beyond).

Recently, Kinoshita [26] showed that a model of "honest-verifier" zero knowledge QMIP can be lifted to general zero knowledge QMIP protocols. He also shows that QMIP have interactive proofs with *computational* zero knowledge proofs under a computational assumption.

Coudron and Slofstra prove a similar result to [19] for multiprover proofs with commuting operator strategies, showing that this class also contains languages of arbitrarily large time complexity, if the promise gap is allowed to be arbitrarily small [16]. Their results (achieved via a completely different method from ours) also show that there are two-prover zero knowledge proofs for languages of arbitrarily large time complexity, albeit in the commuting operator model and with a quantitatively worse lower bound than Corollary 3.

Finally, Crépeau and Yang [17] refined the notion of zero knowledge, requiring the simulator to be local, i.e., that there are non-communicating classical simulators that simulate the (joint) output distribution of the provers. We note that our result does not fulfill this modified definition, and we leave it as an open problem (dis)proving that all MIP* can be made zero knowledge in this setting.

Organization

The paper is organized as follows. We start with some preliminaries in Section 2. Then, in Section 3, we present our transformation on MIP* protocols. In Section 4, we prove the zero knowledge property of the transformed protocol. In Section 5, we prove that the concatenated Steane code is simulatable.

Acknowledgments

AG thanks Thomas Vidick for discussions on related topics. WS thanks Matt Coudron, David Gosset, and Jon Yard for helpful discussions. AG is supported by ERC Consolidator Grant 615307-

[5] An interactive PCP is a protocol where the verifier and a single prover first commit to an oracle, which the verifier can query a bounded number of times. Then, the verifier and prover engage in an interactive proof. An *algebraic* interactive PCP is one where the committed oracle has a desired algebraic structure. We refer to [10] for an in-depth discussion of these models.

QPROGRESS. WS is supported by NSERC DG 2018-03968.

2 Preliminaries

2.1 Notation

We denote $[n]$ as the set $\{1, ..., n\}$. We assume that all Hilbert spaces are finite-dimensional. An n-qubit binary observable (also called a reflection) O is a Hermitian matrix with ± 1 eigenvalues.

We use the terminology "quantum register" to name specific quantum systems. We use sans-serif font to denote registers, such as A, B. For example, "register A", to which is implicitly associated the Hilbert space $\mathcal{H}_{\mathsf{A}}$.

For a density matrix ρ defined on some registers $\mathsf{R}_1 \cdots \mathsf{R}_n$, and a subset S of those registers, we write $\mathrm{Tr}_S(\rho)$ to denote the partial trace of ρ over those registers in S. We write $\mathrm{Tr}_{\overline{S}}(\rho)$ to denote tracing out all registers of ρ *except* for the registers in S.

Let $\sigma_I, \sigma_X, \sigma_Y, \sigma_Z$ denote the four single-qubit Pauli observables

$$\sigma_I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} , \qquad \sigma_X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} , \qquad \sigma_Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} , \qquad \sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} .$$

We let $\mathcal{P}_n$ denote the n-qubit Pauli group, so $\mathcal{P}_n$ is the set of n-qubit unitaries $W_1 \otimes \cdots \otimes W_n$ where $W_i \in \{\pm\sigma_I, \pm i\sigma_I, \pm\sigma_X, \pm i\sigma_X, \pm\sigma_Y, \pm i\sigma_Y, \pm\sigma_Z, \pm i\sigma_Z\}$.

We use two ways of specifying a Pauli observable acting on a specific qubit.

1. Let $W \in \{I, X, Z\}$ be a label and let R be a single-qubit register. We write $\sigma_W(\mathsf{R})$ to denote the observable σ_W acting on R.
2. Let R be an n-qubit register, and let $i \in \{1, \ldots, n\}$. Let $W = X_i$ (resp. $W = Z_i$). We write σ_W to denote the σ_X (resp. σ_Z) operator acting on the i-th qubit in R (the register R is implicit).

We also use W to label Pauli operators that have higher "weight". For example, for $W = X_i Z_j$ the operator σ_W denotes the tensor product $\sigma_{X_i} \otimes \sigma_{Z_j}$.

Universal set of gates A universal set of gates is $\{H, \Lambda(X), \Lambda^2(X)\}$, where H is the Hadamard gate, $\Lambda(X)$ is the controlled-X gate (also known as the CNOT gate), and $\Lambda^2(X)$ is the Toffoli gate [1].

2.2 Error correcting codes

Quantum error correcting codes (QECCs) provide a way of encoding quantum information in a form that is resilient to noise. Specifically, a $[[n, k]]$ quantum code $\mathcal{C}$ encodes all k-qubit states $|\psi\rangle$ into an n-qubit state $\mathsf{Enc}(|\psi\rangle)$. We say that a $[[n, k]]$ QECC has distance d if for any quantum operation $\mathcal{E}$ that acts on at most $(d-1)/2$ qubits, the original state $|\psi\rangle$ can be recovered from $\mathcal{E}(\mathsf{Enc}(|\psi\rangle))$. In this case, we say that $\mathcal{C}$ is a $[[n, k, d]]$ QECC.

Throughout this paper, we mostly use codes that encode 1 logical qubit into some number of physical qubits. If Enc is the encoding map of an $[[m, 1]]$ QECC $\mathcal{C}$ and $|\phi\rangle$ is an n-qubit state, then we overload notation and write $\mathsf{Enc}(|\phi\rangle)$ to denote the mn qubit state obtained from applying Enc to every qubit of $|\phi\rangle$. We refer to the qubits of $|\phi\rangle$ as *logical qubits*, and the qubits of the encoded state $\mathsf{Enc}(|\phi\rangle)$ as *physical qubits*. We call any state $|\psi\rangle$ in the code $\mathcal{C}$ a *codeword*.

Given two QECCs $\mathcal{C}_1$ and $\mathcal{C}_2$, the concatenated code $\mathcal{C}_1 \circ \mathcal{C}_2$ is defined by setting $\mathsf{Enc}_{\mathcal{C}_1 \circ \mathcal{C}_2}(\rho) = \mathsf{Enc}_{\mathcal{C}_2}(\mathsf{Enc}_{\mathcal{C}_1}(\rho))$, i.e. to encode ρ in the concatenated code, we first encode it using $\mathcal{C}_1$, and then encode every physical qubit of $\mathsf{Enc}_{\mathcal{C}_1}(\rho)$ using $\mathcal{C}_2$.

2.2.1 Inner and outer codes

In our zero knowledge transformation, we use quantum error correcting codes in two different ways. One use, as described in the proof overview in Section 1.2, is in the transformation from the original MIP* verifier V to a fault-tolerant version V_{enc}. We call the error correcting code used in the fault tolerant construction the *inner code*, denoted by $\mathcal{C}_{inner}$.

The other use of quantum error correcting codes is in the protocol compression of V_{enc} into the zero knowledge protocol V_{ZK}. In Section 1.2, we described the protocol V_{ZK} as testing whether the players share a history state $|\Phi\rangle$ of the protocol corresponding to V_{enc}. Actually, the protocol tests whether the players share an *encoding* of the history state. The qubits of the history state $|\Phi\rangle$ corresponding to the state of the verifier V_{enc} are supposed to be encoded using another error correcting code and distributed to multiple players (see Section 3.2.1 for more details). For this, we use what we call the *outer code*, denoted by $\mathcal{C}_{outer}$.

The outer code For the outer code $\mathcal{C}_{outer}$, we require a stabilizer code that satisfies the following properties [19]:

1. For every qubit i, there exists a logical X and Z operator that acts trivially on that qubit.
2. The code can correct one erasure in a known location.

The following four-qubit error detection code satisfies both properties [23].

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle) \qquad |1\rangle \mapsto \frac{1}{\sqrt{2}}(|1001\rangle + |0110\rangle)\,.$$

The stabilizer generators for this code are $XXXX, ZIIZ, IZZI$. A set of logical operators for this code are $XIIX, IXXI, ZZII, IIZZ$. We use Enc_{outer} to denote the encoding map for the outer code $\mathcal{C}_{outer}$.

The inner code For the inner code $\mathcal{C}_{inner}$, we use the concatenated Steane code Steane^K for some sufficiently large (but constant) K. We use Enc_{inner} to denote the encoding map for the outer code $\mathcal{C}_{inner}$. We describe the concatenated Steane code in more detail in Section 5.5.1.

2.2.2 Encodings of gates and simulatable codes

An important concept in our work is that of *simulatable codes*. The motivation for this concept is the observation that for a distance d code $\mathcal{C}$, the reduced density matrix of any codeword $|\psi\rangle \in \mathcal{C}$ on fewer than $d-1$ qubits is a state that is independent of $|\psi\rangle$, and only depends on the code $\mathcal{C}$. We generalize this indistinguishability notion to the context of fault tolerant encodings of gates with a QECC: informally, a QECC is simulatable if "small width" reduced density matrices of codewords $|\psi\rangle$ *in the middle* of a logical operation are independent of $|\psi\rangle$. Intuitively, simulatability is a necessary condition for fault tolerant quantum computation; if local views of an in-progress quantum computation are dependent on the logical data, then environmental noise can corrupt the computation.

Let U be a k-qubit gate. If $\underline{a} = (a_1, \ldots, a_k)$ is a k-tuple of distinct numbers between 1 and n, we let $U(\underline{a})$ be the gate U applied to qubits $(a_1, \ldots, a_k)$. If ρ is an n-qubit state, then $U(\underline{a})\rho U(\underline{a})^\dagger$ is the result of applying U to ρ in qubits $a_1, \ldots, a_k$.

An encoding of a k-qubit gate U in the code $\mathcal{C}$ is a way to transform $\mathsf{Enc}(\rho)$ to $\mathsf{Enc}(U(\underline{a})\rho U(\underline{a})^\dagger)$ by applying operations on the physical qubits, sometimes with an additional ancilla state used as a resource. More formally, an *encoding of a k-qubit U in code $\mathcal{C}$* is a pair of states σ_U and σ'_U, and a number $\ell \geq 1$, along with a mapping from k-tuples $\underline{a}$ of distinct physical qubits to sequences of unitaries $O_1(\underline{a}), \ldots, O_\ell(\underline{a})$ such that

$$(O_\ell(\underline{a}) \cdots O_1(\underline{a}))\,(\mathsf{Enc}(\rho) \otimes \sigma_U)\,(O_\ell(\underline{a}) \cdots O_1(\underline{a}))^\dagger = \mathsf{Enc}(U(\underline{a})\rho U(\underline{a})^\dagger) \otimes \sigma'_U,$$

where (in a slight abuse of notation) the unitaries $O_1(\underline{a}), \ldots, O_\ell(\underline{a})$ act only on the physical qubits corresponding to logical qubits $a_1, \ldots, a_k$, as well as the ancilla register holding σ_U. In this definition, the sequence $O_1(\underline{a}), \ldots, O_\ell(\underline{a})$ depends on $\underline{a}$. However, in practice $\underline{a}$ is only used to determine which physical qubits the gates $O_1(\underline{a}), \ldots, O_\ell(\underline{a})$ act on, and otherwise the sequence depends strictly on U. We say that an encoding *uses physical gates $\mathcal{G}$* if for every $\underline{a}$, the unitaries $O_1(\underline{a}), \ldots, O_\ell(\underline{a})$ are gates in $\mathcal{G}$.

If a QECC $\mathcal{C}$ can correct arbitrary errors on s qubits, then the partial trace $\mathrm{Tr}_{\overline{S}}(\mathsf{Enc}(\rho))$ is independent of the state ρ for every set of physical qubits S with $|S| \leq s$. If we start with an encoded state $\mathsf{Enc}(\rho)$, and apply an encoded logical operation U to some k-tuple of qubits $\underline{a}$, then we start in state $\mathsf{Enc}(\rho) \otimes \sigma_U$ and end in state $\mathsf{Enc}(U(\underline{a})\rho U(\underline{a})^\dagger) \otimes \sigma'_U$. So as long as we can compute the partial traces of σ_U and σ'_U, then we can compute $\mathrm{Tr}_{\overline{S}}(\mathsf{Enc}(\rho))$ both before and after the operation. However, the encoded operation is made of up a sequence of gates, and while we are in the middle of applying these gates, the system might not be in an encoded state. We say that an encoding is s-simulatable if we can still compute the reduced density matrices on up to s qubits of the state at any point during the encoding of U. The following definition formalizes this notion:

Definition 5. *An encoding $(\sigma_U, \sigma'_U, \ell, O_1(\underline{a}), \ldots, O_\ell(\underline{a}))$ of a k-qubit gate in a QECC $\mathcal{C}$ is s-*simulatable *if for all integers $0 \leq t \leq \ell$, n-qubit states ρ, and subsets S of the physical qubits of $\mathsf{Enc}(\rho) \otimes \sigma_U$ with $|S| \leq s$, the partial trace*

$$\mathrm{Tr}_{\overline{S}}((O_t(\underline{a}) \cdots O_1(\underline{a}))\mathsf{Enc}(\rho) \otimes \sigma_U (O_t(\underline{a}) \cdots O_1(\underline{a}))^\dagger)$$

can be computed in polynomial time from t, $\underline{a}$, and S. In particular, the partial trace is independent of ρ.

In our applications, s will be constant. We also consider only a finite number of gates U, and since t is bounded in any given encoding, t will also be constant. The partial trace in the above definition will be a $2^{|S|} \times 2^{|S|}$ matrix, where $|S| \leq s$. So when we say that the partial trace can be computed in polynomial time in Definition 5, we mean that the entries of this matrix are rational, and can be computed explicitly in polynomial time from $\underline{a}$, S, and t.

A crucial component of our zero knowledge arguments is the notion of simulatable codes. We state now the theorem we will use to prove zero knowledge. The proof is deferred to Section 5.

Theorem 6. *Let $\mathcal{U} = \{H, \Lambda(X), \Lambda^2(X)\}$. For every constant s, there exists a $[[n, 1]]$ QECC $\mathcal{C}$ where n is constant, such that $\mathcal{C}$ has s-simulatable encodings of $\mathcal{U}$ using only $\mathcal{U}$ as physical gates.*

If a code $\mathcal{C}$ admits a simulatable encoding of a gate U, then, applying Definition 5 with $t = 0$, we see that it must be possible to compute the partial trace $\mathrm{Tr}_{\overline{S}}(\mathsf{Enc}(\rho) \otimes \sigma_U)$ for any set of physical qubits S with $|S| \leq s$, with no knowledge of ρ. In particular, it must be possible to compute partial traces of $\mathsf{Enc}(\rho)$ on all but s qubits. We must also be able to compute the partial traces of the ancilla states σ_U and (setting $t = \ell$) σ'_U, although this is easier in principle, since we have full knowledge of these states.

2.3 Quantum interactive protocols

We first define the notion of a *protocol circuit*, which is a quantum circuit representation an interaction between a quantum verifier and one or more provers. A protocol circuit C with k provers and r rounds is specified by a tuple (n, m, Γ) where n, m are positive integers and Γ is a sequence of gates $(g_1, g_2, \ldots)$. This tuple is interpreted in the following manner. The circuit C acts on these registers:

1. A set of *prover* registers $\mathsf{P}_1, \ldots, \mathsf{P}_k$.
2. A set of *message* registers $\mathsf{M}_1, \ldots, \mathsf{M}_k$; each register M_i consists of m qubits. The j'th qubit of register M_i is denoted by M_{ij}.
3. A *verifier* register V which consists of n qubits. The j'th qubit of register V is denoted by V_j.

Each gate g_i consists of a *gate type*, and the label of the registers that the gate acts on. There are two gate types:

1. A gate from a universal gate set (such as Hadamard, CNOT, and Toffoli), which can only act on registers $\mathsf{V}, \mathsf{M}_1, \ldots, \mathsf{M}_k$.
2. A *prover gate* P_{ij}, which represents the i'th prover's unitary in round j. The prover gate P_{ij} can only act on registers $\mathsf{P}_i\mathsf{M}_i$.

Furthermore, prover i's gates $\{P_{ij}\}$ must appear in order; for example, P_{i2} can only appear in the circuit after P_{i1} has appeared. A prover gate P_{ij} cannot appear twice in the circuit with the same label.

Intuitively, a protocol circuit describes an interaction between a verifier and k provers where the verifier performs a computation on the workspace register V, and communicates with the provers through the message registers $\{\mathsf{M}_i\}$, and the provers carry out their computations on the registers $\{\mathsf{P}_i\mathsf{M}_i\}$. The verifier's workspace V is initialized in the all zeroes state, and the $\{\mathsf{P}_i\mathsf{M}_i\}$ registers are initialized in some entangled state $|\psi\rangle$ chosen by the provers. At the end of the protocol circuit, the first qubit of the workspace register V is measured in the standard basis to determine whether the verifier accepts or rejects.

A *prover strategy* $\mathcal{S}$ for a protocol circuit C is specified by a tuple $(d, \{P_{ij}\}, |\psi\rangle)$ where d is a positive integer, a set of unitary operators P_{ij} for $i = 1, \ldots, k$ and $j = 1, \ldots, r$ that act on $\mathbb{C}^d \otimes (\mathbb{C}^2)^{\otimes m}$, and pure states $|\psi\rangle$ in $(\mathbb{C}^d)^{\otimes k} \otimes (\mathbb{C}^2)^{\otimes mk}$. Given a protocol circuit C, we write $\omega^*(C)$ to denote the supremum of acceptance probabilities of the verifier over all possible prover strategies $\mathcal{S}$.

We now define the complexity class QMIP, which stands for *quantum multiprover interactive proofs*. This is the set of all languages L that can be decided by a quantum interactive protocol with at most polynomially many provers, at most polynomially-many rounds, and polynomial-sized protocol circuits, whose gates are drawn from the gate set $\{H, \Lambda(X), \Lambda^2(X)\}$.

Definition 7. *A promise problem $L = (L_{yes}, L_{no})$ is in the complexity class $\mathsf{QMIP}_{c,s}[k, r]$ if and only if there exists a polynomial-time computable function V with the following properties:*

1. *For every $x \in L_{yes} \cup L_{no}$, the output of V on input x is a description of a k-prover, r-round prover circuit $V(x) = (n, m, \Gamma)$ where $n, m = \mathrm{poly}(|x|)$.*
2. Completeness. *For every $x \in L_{yes}$, it holds that $\omega^*(V(x)) \geq c$.*

3. Soundness. *For every $x \in L_{no}$, it holds that $\omega^*(V(x)) < s$.*

Furthermore, we say that L has a $\mathsf{QMIP}_{c,s}[k,r]$ *protocol V.*

Throughout this paper, we interchangeably refer to $V(x)$ as the protocol circuit, the protocol, or the verifier that is executing the protocol, depending on the context.

We note that in the negative case (i.e. $x \in L_{no}$), we require that the entangled value of $V(x)$ is *strictly* less than s. This allows us to meaningfully talk about "zero promise gap" classes such as $\mathsf{QMIP}_{1,1}[k,r]$, where in the Completeness case, the verifier has to accept with probability 1, whereas in the Soundness case, the verifier has to reject with some positive probability. Finally, we follow the convention that $\mathsf{QMIP}[k,r]$ is defined as $\mathsf{QMIP}_{\frac{2}{3},\frac{1}{3}}[k,r]$.

We also define the class MIP^*, which is defined in the same way as QMIP except that the protocol is specified as a classical interaction between a randomized verifier (modelled as a probabilistic polynomial-time Turing machine) and quantum provers. Since the verifier is classical, the communication between the verifier and provers can be treated as classical. Thus, in a k-prover MIP^* protocol, we can equivalently talk about *measurement prover strategies* $\mathcal{S}$, where the k provers share an entangled state $|\psi\rangle \in \mathcal{H}^{\otimes k}$ for some Hilbert space $\mathcal{H}$. In each round of the protocol, each prover receives a classical message from the verifier, and performs a measurement on their share of $|\psi\rangle$ that depends on the verifier's message as well as the previous messages exchanged between that prover and the verifier (but not the communication with the other provers).

We call prover strategies for a general QMIP protocol as *unitary strategies*, to distinguish them from measurement strategies for MIP^* protocols. Furthermore, when we speak of an MIP^* protocol V, we are referring to the verifier for the protocol (which is some probabilistic Turing machine).

2.4 Zero knowledge MIP^*

First, we define the *view* of an interaction between a classical, randomized verifier $\widehat{V}$ and a set of k provers that behave according to some strategy $\mathcal{S}$, as might occur in an MIP^* protocol. The view is a random variable $\mathrm{View}(\widehat{V}(x) \leftrightarrow \mathcal{S})$ which is the tuple $(x, r, m_1, m_2, \ldots, m_{2r})$ where x is the input to $\widehat{V}$, r is the randomness used by $\widehat{V}$, and the m_i's are the messages between the provers and verifier.

Next, we present the definition of zero knowledge MIP^* protocols, first defined by [11]. We use the abbreviation "PPT" to denote "probabilistic polynomial-time."

Definition 8. *An* $\mathsf{MIP}^*_{c,s}[k,r]$ *protocol V for a promise language $L = (L_{yes}, L_{no})$ is* statistically zero knowledge *if for all $x \in L_{yes}$, there exists a prover strategy $\mathcal{S}$ (called the* honest strategy*) satisfying the following properties:*

1. *The strategy $\mathcal{S}$ is accepted by the protocol $V(x)$ with probability at least c,*
2. *For all PPT verifiers $\widehat{V}$, there exists a PPT simulator $\mathrm{Sim}_{\widehat{V}}$ such that the output distribution of $\mathrm{Sim}_{\widehat{V}}(x)$ is $\varepsilon(n)$-close in total variation distance to* $\mathrm{View}(\widehat{V}(x) \leftrightarrow \mathcal{S})$*, for some negligible function $\varepsilon(n)$.*

Furthermore, the complexity class $\mathsf{SZK\text{-}MIP}^*_{c,s}[k,r]$ *is the set of languages that have statistical zero knowledge proof systems.*

When a language can be decided by a zero knowledge proof system with closeness $\varepsilon(n) = 0$, we say that it admits a *perfect zero knowledge* proof system. In other words, the interaction can be simulated exactly. We let $\mathsf{PZK\text{-}MIP}^*_{c,s}[k,r]$ denote languages that admit perfect zero knowledge MIP^* protocols.

Some subtleties We address two subtleties regarding the definitions of QMIP and PZK-MIP*.

1. The definition of QMIP depends on our choice of gate set. If we allow the verifier circuits to use arbitrary single- and two-qubit gates, then our perfect zero knowledge results may not hold; however, we will still get the statistical zero knowledge property with exponentially small error.

2. In a PZK-MIP$^*_{c,s}[k,r]$ protocol V, there may be no strategy $\mathcal{S}$ for the provers that gets accepted with probability c exactly. Instead, there may be a sequence of strategies whose success probability converges to c. Thus, in order for PZK-MIP$^*_{c,s}[k,r]$ to be correctly defined, we require that there exists a sequence of honest strategies $\mathcal{S}_1, \mathcal{S}_2, \ldots$ satisfying:

    - The success probability of $\mathcal{S}_i$ approaches c as $i \to \infty$, and
    - For all verifiers $\widehat{V}$, there exists a simulator $\mathrm{Sim}_{\widehat{V}}$ whose output distribution can be approximated arbitrarily well by the sequence of honest strategies. In other words, for all δ there exists an i such that the total variation distance between $\mathrm{View}(\widehat{V}(x) \leftrightarrow \mathcal{S}_i)$ and $\mathrm{Sim}_{\widehat{V}}$ is at most δ.

    This subtlety only arises when considering "zero gap" classes such as PZK-MIP$^*_{1,1}[k,r]$.

2.5 Parallel repetition

Parallel repetition of interactive protocols is a commonly used technique for performing *gap amplification*. We now define what this means for 1-round MIP* protocols.

Definition 9 (Parallel repetition of a one-round MIP* protocol)**.** *Let V denote a 1-round, k-prover* MIP* *protocol. The m-fold parallel repetition of V is another 1-round, k-prover* MIP* *protocol V^m where m independent instances of V are executed simultaneously. Let q_{ij} denote the questions from instance i to prover j. Then prover j receives $(q_{1j}, q_{2j}, \ldots, q_{mj})$ simultaneously, and responds with answers $(a_{1j}, a_{2j}, \ldots, a_{nj})$. The answers $(a_{i1}, a_{i2}, \ldots, a_{ik})$ is then given to the i'th verifier instance, and V^m accepts if and only if all instances accept.*

The behaviour of $\omega^*(V^m)$ as a function of n and $\omega^*(V) < 1$ is non-trivial; clearly, if $\omega^*(V) = 1$, then $\omega^*(V^m) = 1$ as well. Although one might expect that $\omega^*(V^m)$ decays exponentially with m in the case that $\omega^*(V) < 1$, this is not known in general. Raz [34] showed that such exponential decay *does* hold for classical 1-round, 2-prover MIP proof systems, but extending this to the case of more provers or MIP* proof systems has remained an active area of research. It is an open question for whether the analogue of Raz's result holds for MIP* protocols (although a polynomial-decay bound is known [40]).

Bavarian, Vidick, and Yuen [3] showed that an exponential-decay parallel repetition theorem also holds for 1-round MIP* protocols that have the property of being *anchored*, and furthermore, *every* 1-round MIP* protocol can be transformed into an equivalent anchored protocol. Their result has the additional benefit in that it holds for any number of provers.

We do not formally define the anchoring property here, but instead we describe a simple transformation to anchor any 1-round MIP* protocol.

Definition 10 (Anchoring)**.** *Let $\alpha > 0$ be some constant. Given a 1-round, k-prover* MIP* *protocol V, define its α-*anchored version *$V_\perp$ to be the protocol which:*

1. *Runs the verifier in V to obtain questions $(q_1, \ldots, q_k)$ for the k provers.*

2. *Independently choose each coordinate i with probability α and replace q_i with an auxiliary question symbol $\perp$, and send the questions to each prover.*

3. *If any prover received the auxiliary question $\perp$, automatically accept. Otherwise, accept the provers' answers only if V would have accepted.*

This transformation preserves completeness and soundness: $\omega^*(V) = 1$ if and only if $\omega^*(V_\perp) = 1$. In general, we have the relationship

$$\omega^*(V_\perp) = (1-\alpha)^k \omega^*(V) + (1-(1-\alpha)^k).$$

Bavarian, Vidick and Yuen [3] showed the parallel repetition of anchored games admits an exponential decay in success probability.

Theorem 11. *Let $\alpha > 0$. Let V be a 1-round, k-prover MIP^* protocol. Let $V_\perp$ be the α-anchored version of V as defined in Definition 10. Let $m > 0$ be an integer. If $\omega^*(V) = 1$, then $\omega^*(V_\perp^m) = 1$. Otherwise,*

$$\omega^*(V_\perp^m) \leq \exp(-\beta \varepsilon^\gamma m)$$

where β is a universal constant depending on α and the protocol V, ε is defined as $1 - \omega^*(V)$, and γ is a universal constant.

3 Our zero knowledge protocol

In this section we present the zero knowledge transformation for general MIP^* protocols. For convenience we reproduce the statement of Theorem 1.

Theorem 1. *For all $0 \leq s \leq 1$, for all polynomially bounded functions k, r,*

$$\mathsf{MIP}^*_{1,s}[k,r] \subseteq \mathsf{PZK\text{-}MIP}^*_{1,s'}[k+4,1]$$

where $s' = 1-(1-s)^\alpha$ for some universal constant $\alpha > 0$.

Fix a promise language $L \in \mathsf{MIP}^*_{1,s}[k,r]$. There exists a polynomial-time computable function V that on input x outputs a k-prover, 1-round protocol circuit $V(x)$ such that if $x \in L$, then $\omega^*(V(x)) = 1$, and otherwise $\omega^*(V(x)) < s$. Furthermore, since we are dealing with an MIP^* proof system, the communication between the verifier and the provers is classical. Thus, we can assume that the protocol circuit V has the following structure. All qubits of the verifier register V are initialized to $|0\rangle$. The protocol circuit proceeds in five phases:

- **Verifier Operation Phase 1**: All computation in this phase of the protocol occurs on the verifier register V. At the end of the computation, the verifier's messages to the i'th prover are stored in a subregister N_i of V.

- **Copy Question Phase**: For each prover i, CNOT gates are applied bitwise from N_i to bits in the register M_i.

- **Prover Operation Phase**: Each prover i applies prover gate P_i to registers $\mathsf{P}_i\mathsf{M}_i$, in sequence.

- **Copy Answer Phase**: For each prover i, CNOT gates are applied bitwise from M_i to bits in the register N_i.

- **Verifier Operation Phase 2**: The remaining computation in the protocol occurs on the verifier register V, and the accept/reject decision bit is stored in a designated output qubit of V.

As mentioned earlier, we assume that the non-prover gates of the protocol circuit $V(x)$ are drawn from the universal gate set $\{H, \Lambda(X), \Lambda^2(X)\}$. Figure 1 gives a diagrammatic representation of this five-phase structure, depicting a protocol in which a verifier interacts with a single prover.

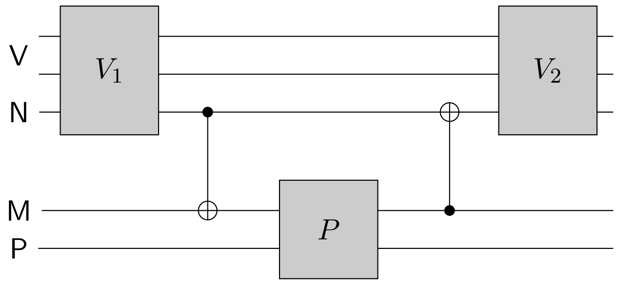


Figure 1: A quantum circuit representation of an MIP* protocol

As described in the Introduction, we first transform the protocol circuit $V(x)$ into an equivalent protocol circuit $V_{enc}(x)$ that performs its computations fault-tolerantly. Then, we use the compression techniques of [25, 19] on the protocol defined by $V_{enc}(x)$ to obtain a protocol $V_{ZK}(x)$ which has the desired zero knowledge properties.

3.1 Robustifying protocol circuits

We now describe a polynomial-time transformation that takes as input the description of a k-prover, 1-round MIP* protocol circuit such as V described above, and outputs another k-prover, 1-round protocol circuit V_{enc} that describes an *equivalent* MIP* protocol, but has additional fault-tolerance properties.

The non-prover gates of V_{enc} are drawn from the universal gate set $\{H \otimes H, \Lambda(X), \Lambda^2(X)\}$.[6] The registers that are involved in the protocol V_{enc} are $\{\mathsf{P}_1, \ldots, \mathsf{P}_k, \mathsf{M}_1, \ldots, \mathsf{M}_k, \mathsf{V}\}$. The verifier workspace register V can be subdivided into registers A, B, O, and $\mathsf{N}_1, \ldots, \mathsf{N}_k$. Intuitively, the register A holds encoded qubits, the register B holds unencoded qubits, the register O holds an encoding of the output bit at the end of the protocol, and the register N_i is isomorphic to M_i for all i.

Let the inner code $\mathcal{C}_{inner}$ be a 192-simulatable code. We remark that from Theorem 6, such codes exist and each logical qubit is encoded in m physical bits, for some constant m.

At the beginning of the protocol V_{enc}, the qubits in register V are initialized to zero. In addition to the five phases of V, there are two additional phases in V_{enc}. First, the protocol V_{enc} goes through a **Resource Generation Phase**, in which the verifier generates many $\mathcal{C}_{inner}$ encodings of the following states in its private workspace:

1. Toffoli magic states $|\mathsf{Toffoli}\rangle = \Lambda^2(X)(H \otimes H \otimes I)|0, 0, 0\rangle$.
2. Ancilla $|0\rangle$ qubits.
3. Ancilla $|1\rangle$ qubits.

[6]The doubled Hadamard gate is used for technical reasons; the second Hadamard gate can always be applied to unused ancilla qubits if it is not needed.

Thus the state of the register V after the Resource Generation Phase will be a tensor product of encoded magic states, encoded $|0\rangle$ states, encoded $|1\rangle$ states, and unencoded $|0\rangle$ states.

Now the the verifier of V_{enc} simulates the five computational phases of V, but as logical operations acting on data encoded using the inner code $\mathcal{C}_{inner}$. For the Verifier Operation Phases and the Copy Question/Answer Phases, each non-prover gate $g_i \in \{H, \Lambda(X), \Lambda^2(X)\}$ of V is replaced in V_{enc} with the encoding of g_i using the $\mathcal{C}_{inner}$, as given by Theorem 6. For example, if g_i in V is a Hadamard gate that acts on some qubit α of V, then its equivalent will be a sequence of (double) Hadamard gates acting transversally on the physical qubits of the encoding of qubit α. If g_i in V is a Toffoli gate, then in V_{enc} the logical gate is applied using the Toffoli gadget (as described in Section 5). Thus, all of the gates of the verifier in V are performed in an encoded manner in V_{enc}.

The Prover Operation Phase proceeds as before; each prover applies their prover gate on the MP registers in sequence. We assume that the Prover Operation Phase is padded with sufficiently many identity gates so that the number of time steps in between each prover gate application is at some sufficiently large constant times the block length of the inner code $\mathcal{C}_{inner}$.

Note that the questions to the provers are encoded using the inner code $\mathcal{C}_{inner}$; this is not a problem for the provers, who can decode the questions before performing their original strategy, and encode their answers afterwards.

Finally, we assume that at the end of the (encoded) Verifier Operation Phase 2, the register O stores the logical encoding of the accept/reject decision bit. After Verifier Operation Phase 2, the protocol V_{enc} executes the **Output Decoding Phase**, where the logical state in register O is decoded (using the decoder from $\mathcal{C}_{inner}$) into a single physical qubit O_{out}.

It is easy to see that this transformation from V to V_{enc} preserves the acceptance probability of the protocol.

Proposition 12. *For all* 1*-round* MIP^* *protocols* V*, for the* MIP^* *protocol* V_{enc} *that is the result of the transformation just described, we have that*

$$\omega^*(V) = \omega^*(V_{enc}).$$

3.1.1 Micro-phases of V_{enc}

We assume the following structural format to the protocol circuit V_{enc}: aside from the major phases of V_{enc}, we can partition the timesteps of the circuit into "micro-phases", where each micro-phase consists of a constant number of consecutive timesteps, and each micro-phase can be classified according to the operations performed within it:

- **Idling**: the gates applied by the verifier during this micro-phase are all identity gates.
- **Resource encoding**: gates are applied to a collection of ancilla $|0\rangle$ qubits to form either an encoding of a $|0\rangle$ state, $|1\rangle$ state, or a Toffoli magic state.
- **Logical operation**: the encoding of a single logical gate is being applied to some encoded blocks of qubits, possibly along with some unencoded ancilla qubits.
- **Output decoding**: the output register O of the verifier circuit is decoded to obtain a single qubit answer. This is exactly the Output Decoding phase.

For example, the Resource Generation phase consists of a sequence of resource encoding micro-phases, applied to blocks of ancilla qubits. The Verifier Operation phases consist of sequences of both idling steps and logical operations, applied to blocks of encoded qubits as well as ancilla qubits. The timesteps during the Prover Operation phase are classified as idling steps, because the verifier is not applying any gates to its private space.

3.1.2 Prover reflection times

Given the protocol circuit V_{enc} of length T, we identify special timesteps during the protocol corresponding to the timesteps where the provers apply their prover gate. For every prover i, we define $t_\star(i) \in \{0, 1, 2, \ldots, T\}$ to be the time in the protocol circuit when prover i applies their prover gate P_i.

3.2 A zero knowledge MIP^* protocol to decide L

Given the transformation from an MIP^* protocol V to an equivalent "fault-tolerant" protocol V_{enc}, we now introduce a second transformation that takes V_{enc} and produces another equivalent MIP^* protocol V_{ZK} that has the desired zero knowledge properties.

This protocol is obtained by applying the compression procedure of [25, 19] to V_{enc}. Since we are compressing interactive protocols (involving verifiers and provers) into other interactive protocols, to keep things clear we use the following naming convention:

- **Verifiers** and **provers** refer to the parties in V_{enc} (i.e. the protocol that is being *compressed*);
- **Referees** and **players** refer to the parties in V_{ZK} (i.e. the protocol that is the result of the compression scheme).

At a high level, the protocol V_{ZK} is designed to verify that the players possess (an encoding of) a *history state* of the protocol V_{enc}:

$$|\Phi\rangle_{\mathsf{CVMP}} = \frac{1}{\sqrt{T+1}} \sum_{t=0}^{T} |\mathsf{unary}(t)\rangle_{\mathsf{C}} \otimes |\Phi_t\rangle_{\mathsf{VMP}} \tag{2}$$

where T is the number of gates of V_{enc}, $\mathsf{unary}(t) = t_1 t_2 \cdots t_T$ denotes the unary encoding of time t, i.e.

$$t_\ell = \begin{cases} 1 & \text{if } \ell \le t \\ 0 & \text{otherwise} \end{cases},$$

and $|\Phi_t\rangle$ is the state of the protocol V_{enc} after t time steps (called the t'th *snapshot state*).

We specify some details of the protocol V_{ZK}:

- **Rounds**: 1-round protocol
- **Number of players**: $k+4$ players, which are are divided into k *prover players* (labelled $PP_1, \ldots, PP_k$) and 4 *verifier players* (labelled $PV_1, \ldots, PV_4$).
- **Question and answer format**: questions to the verifier players are 6-tuples of the form $(W_1, \ldots, W_6)$, where each W_i denotes a two-qubit Pauli observable on some specified pair of qubits, and the six observables commute. Furthermore, the Pauli observables are tensor products of operators from the set $\{I, X, Z\}$. An example of a question would be: $(X_1X_2, Z_1Z_2, I_7Z_5, X_3Z_4, Z_3X_4, X_7I_5)$. Verifier players' answers are a 6-tuple of bits $(a_1, \ldots, a_6)$.

  Questions to prover player PP_i can be one of three types:

  1. Prover reflection, denoted by $\star_i$.
  2. Question gates, denoted by Q_{ij} for $j = 1, \ldots, m'$, where m' is the maximum number of qubits in the message registers $\{\mathsf{M}_i\}$ in the protocol V_{enc}.

3. Question flag flip, denoted by QF_i.
4. Answer gates, denoted by A_{ij} for $j = 1, \dots, m'$.
5. Answer flag flip, denoted by AF_i.

We notice that even if the Prover players' original answers consisted of a single bit, after robustifying the protocol circuits, the answers become an encoding of the logical bit.

The distribution of questions and the rules used by the referee in V_{ZK} are essentially identical to the ones used in the compression protocol in [19].[7] Given those, the results of [19] show that V_{ZK} is a complete and sound MIP^* proof system for L:

$$L \in \mathsf{MIP}^*_{1,s'}[k+4, 1]$$

where $s' = (1-s)^\beta / p(n)$ for some universal constant β and some polynomial $p(n)$ that depends on the original protocol V, and s is the soundness of V.

The details of the the question distribution, the rules and the soundness analysis are irrelevant for this paper, as we are only concerned with establishing the zero knowledge property of V_{ZK}. For this, we only need to consider the interaction between honest players and a potentially cheating referee $\widehat{R}$.

3.2.1 An honest strategy $\mathcal{S}_{ZK}$ for V_{ZK}

We now specify an honest strategy $\mathcal{S}_{ZK}(x)$ for the players in $V_{ZK}(x)$ in the case that $x \in L_{yes}$. Since $x \in L_{yes}$, by definition we have that $\omega^*(V(x)) = 1$, and therefore by Proposition 12 we get $\omega^*(V_{enc}(x)) = 1$. Thus there exists a sequence of finite dimensional unitary strategies $\{\mathcal{S}_1(x), \mathcal{S}_2(x), \dots\}$ for $V_{enc}(x)$ such that the acceptance probability approaches 1. For simplicity, we assume that there exists a finite dimensional unitary strategy $\mathcal{S}(x)$ for $V_{enc}(x)$ that is accepted with probability 1; in the general case, we can take a limit and our conclusions still hold.

The strategy $\mathcal{S}(x)$ consists of a dimension d, an entangled state $|\psi\rangle$ on registers $\mathsf{P}_1, \dots, \mathsf{P}_k$ and $\mathsf{M}_1, \dots, \mathsf{M}_k$ (where the registers P_i have dimension d), and a collection of unitaries $\{P_i\}$ where P_i acts on registers $\mathsf{P}_i\mathsf{M}_i$. We assume, without loss of generality, that in under the strategy $\mathcal{S}$ in protocol $V_{enc}(x)$, the state of the message registers $\{\mathsf{M}_i\}_i$ are in the code subspace of $\mathcal{C}_{inner}$ at each time step of the protocol (where we treat the prover operations as taking one time step).

Given this, we define the measurement[8] strategy $\mathcal{S}_{ZK}(x)$ in the following way. For notational simplicity, we omit mention of the input x when it is clear from context.

The shared entanglement Let $|\Phi\rangle_{\mathsf{CVMP}}$ denote the history state of the protocol $V_{enc}(x)$ when the provers use strategy $\mathcal{S}$ (as in (2)). The initial state $|\Phi_0\rangle$ is $|0\rangle_{\mathsf{V}} \otimes |\psi\rangle_{\mathsf{MP}}$.

We now construct an *distributed history state* $|\Phi'\rangle_{\mathsf{C'V'MPF}}$ from $|\Phi\rangle$ in two steps. First, without loss of generality we augment a k-partite register $\mathsf{F} = \mathsf{F}_1, \dots, \mathsf{F}_k$ to $|\Phi\rangle$ so that serves as flags that indicate which operations the i'th prover has applied. Thus the augmented history state looks like

$$|\Phi\rangle = \frac{1}{\sqrt{T+1}} \sum_{t=0}^{T} |\mathsf{unary}(t)\rangle_{\mathsf{C}} \otimes |\Delta_t\rangle_{\mathsf{VMP}} \otimes |f(t)\rangle_{\mathsf{F}}$$

[7] The main difference concerns the questions "Question flag flip" and "Answer flag flip" to the provers, which do not occur in [25, 19]. These will be helpful for the analysis of zero knowledge property. We explain in Appendix A the slight modifications to the protocol from [19] that are needed.

[8] Since the protocols V and V_{enc} are general QMIP protocols, the strategy $\mathcal{S}$ is a unitary strategy. Since V_{ZK} is a MIP^* protocol, we specify $\mathcal{S}_{ZK}$ as a *measurement* strategy.

where $|f(t)\rangle_{\mathsf{F}} = \bigotimes_i |f_i(t)\rangle_{\mathsf{F}_i}$ and $|f_i(t)\rangle_{\mathsf{F}_i} = |q_i(t)\rangle_{\mathsf{F}_{Q_i}} \otimes |p_i(t)\rangle_{\mathsf{F}_{P_i}} \otimes |a_i(t)\rangle_{\mathsf{F}_{A_i}}$. For all $i \in \{1, 2, \ldots, k\}$, the functions $q_i(t), p_i(t), a_i(t)$ are boolean functions of the time t, defined as follows:

$$q_i(t) = \begin{cases} 1 & \text{if } t \geq t_\star(i) - 1 \\ 0 & \text{otherwise} \end{cases},$$

$$p_i(t) = \begin{cases} 1 & \text{if } t \geq t_\star(i) \\ 0 & \text{otherwise} \end{cases},$$

and

$$a_i(t) = \begin{cases} 1 & \text{if } t \geq t_\star(i) + 1 \\ 0 & \text{otherwise} \end{cases}.$$

The flags q_i, p_i, a_i flip from 0 to 1 consecutively: at time $t = t_\star(i) - 2$, all flags for player i are set to 0. By the time $t = t_\star(i) + 1$, all flags for player i are set to 1.

Next, we perform a qubit-by-qubit encoding of the C and V registers of $|\Phi\rangle$ using the *outer code* $\mathcal{C}_{outer}$, to obtain the *encoded history state* $|\Phi'\rangle$ defined on registers $\mathsf{C}', \mathsf{V}', \mathsf{M}, \mathsf{P}$. Each qubit of C and V are encoded into 4 physical qubits.

The allocation of the registers of $|\Phi'\rangle$ to the $k+4$ players are as follows:

1. The register C consists of T qubits. For $i = 1, \ldots, T$, let C_i denote the i'th qubit register of C. For $j = 1, \ldots, 4$, let C'_{ij} denote the j'th share of the $\mathcal{C}_{outer}$ encoding of C_i. In the honest case, the j'th verifier player PV_j has the qubits $\{\mathsf{C}'_{ij}\}_i$.

2. Similarly, the registers V'_{ij} denote the j'th share of the encoding of the register V_i; the subregisters $\mathsf{A}_i, \mathsf{B}_i, \mathsf{O}_i, \mathsf{N}_i$ of V are encoded into subregisters $\mathsf{A}'_{ij}, \mathsf{B}'_{ij}, \mathsf{O}'_{ij}, \mathsf{N}'_{ij}$ of V' respectively. In the honest case, the j'th verifier player PV_j holds qubits $\{\mathsf{V}'_{ij}\}_i$.

3. The prover players' $\{PP_1, \ldots, PP_k\}$ represent the original k players of the protocol V and V_{enc}. In the honest case, prover player PP_i holds registers $\{\mathsf{F}_i \mathsf{P}_i \mathsf{M}_i\}$. Note that these registers are not encoded and split up like with the clock and verifier registers.

Player measurements Since V_{ZK} is a 1-round MIP^* protocol, we specify the strategy $\mathcal{S}_{ZK}$ in terms of measurement operators.

- When the verifier players receive a 6-tuple of commuting Pauli observables $(W_1, \ldots, W_6)$, they measure each of the observables $\sigma_{W_1}, \ldots, \sigma_{W_6}$ in sequence on the designated qubits of their share of $|\Phi'\rangle$, and report the measurement outcomes $(a_1, \ldots, a_6)$. For example, if $W_1 = X_1 Z_2$, then the corresponding observable would be $\sigma_X \otimes \sigma_Z$ acting on qubits labelled 1 and 2.

- When prover player PP_i receives a prover reflection question $\star_i$, they measure the following observable on the registers $\mathsf{F}_{P_i} \mathsf{P}_i \mathsf{M}_i$:

$$P'_i = |0\rangle\langle 1|_{\mathsf{F}_{P_i}} \otimes P_i^\dagger + |1\rangle\langle 0|_{\mathsf{F}_{P_i}} \otimes P_i$$

where P_i acts on $\mathsf{P}_i \mathsf{M}_i$. It is easy to see that P'_i is an observable with a $+1$ eigenspace and a -1 eigenspace.

- When prover player PP_i receives a "Question gate" question Q_{ij}, they measure the observable σ_X on the register M_{ij}, and report the one-bit answer. When PP_i receives an "Answer gate" question A_{ij}, they measure the observable σ_Z on the register M_{ij}, and report the one-bit answer.

- When prover player PP_i receives the "Question flag flip" question QF_i, they measure the observable σ_X on the register F_{Q_i}. When they receive "Answer flag flip" question AF_i, they measure the observable σ_X on the register F_{A_i}.

The analysis of the compression protocol in [19] implies that the strategy $\mathcal{S}_{ZK}(x)$ is accepted in the protocol $V_{ZK}(x)$ with probability 1. We now proceed to argue the zero knowledge property of the protocol V_{ZK} with the honest player strategy $\mathcal{S}_{ZK}$.

Notation	**Meaning**
V_{enc}	The fault tolerant encoding of the original protocol V
k	Number of provers in the protocol V_{enc}
V_{ZK}	The zero knowledge protocol
N_V	Number of verifier players in V_{ZK}, which is 4.
$t_\star(i)$	The time that prover i applies prover reflection P_i'.
$q_i(t)$	Indicator function that is 1 iff $t \geq t_\star(i) - 1$. Used as a flag to indicate whether the questions for the i'th prover have been all copied.
$p_i(t)$	Indicator function that is 1 iff $t \geq t_\star(i)$. Used as a flag to indicate whether the i'th prover has applied its reflection P_i'.
$a_i(t)$	Indicator function that is 1 iff $t \geq t_\star(i) + 1$. Used as a flag to indicate whether the i'th prover is ready to copy its answers to the verifier.
$\mathcal{S}_{ZK}$	The honest player strategy for V_{ZK}
$\|\Phi\rangle$	The unencoded history state of an interaction in the protocol V_{enc}
$\|\Phi_I\rangle$	The restriction of the history state $\|\Phi\rangle$ to a time interval I
P_i'	The prover reflection used by prover player PP_i in $\mathcal{S}_{ZK}$
$\widehat{R}$	A (possibly cheating) referee in the protocol V_{ZK}
$\widehat{W}$	A tuple of questions in V_{ZK}, or the associated observable measured by the players in $\mathcal{S}_{ZK}$.
$\widehat{W}^{(V,r)}, \widehat{W}^{(P,r)}$	Questions to the r'th verifier and prover players, respectively.
$\widetilde{W}^{(V,r)}, \widetilde{W}^{(P,r)}$	Players' measurement observables without the prover reflections
L	This is the number of verifier player qubits that can be addressed by a question $\widehat{W}$. This is $12N_V = 48$, which is a constant.

Figure 2: Notation reference

4 Zero knowledge property of V_{ZK}

Let $\widehat{R}(x)$ be an arbitrary referee (modelled as a probabilistic polynomial-time Turing machine) interacting with k provers that use the measurement strategy $\mathcal{S}_{ZK}(x)$ defined above. In general, this referee $\widehat{R}(x)$ may try to gain forbidden knowledge by deviating from the behaviour of the referee specified by the protocol V_{ZK}. In this section, we show this cannot happen by describing an efficient simulator $\mathrm{Sim}(x)$ whose output distribution is equal to $\mathrm{View}(\widehat{R}(x) \leftrightarrow \mathcal{S}_{ZK}(x))$.

A referee $\widehat{R}$ could try to cheat by sampling questions from a different distribution than the one that is specified in the V_{ZK} protocol. Furthermore, the referee could interact with the provers *adaptively*: it could send some messages to a subset of the provers, get some answers, and depending on those responses choose questions for another set of provers. We can assume that a cheating referee does not interact with the same prover in V_{ZK} twice; since the protocol is supposed to be one round, an honest prover would abort the protocol if the referee interacted with it multiple times. Similarly, we assume that a cheating referee only asks questions that match the *format* of questions in V_{ZK}.

In Section Section 4.1, we show how to simulate the interaction between $\widehat{R}$ and the players when $\widehat{R}$ is *non-adaptive*, meaning that the questions for all players are picked simultaneously by the referee before interacting with them. In Section 4.2 we show how to perform this simulation for general adaptive referees $\widehat{R}$.

For the remainder of this section, we omit mention of the input x; we assume that the referee $\widehat{R}$ and the strategy $\mathcal{S}_{ZK}$ implicitly depend on x.

We introduce some additional notation.

- Let $N_V = 4, N_P = k$ denote the number of verifier players and prover players, respectively.
- Let $\widehat{W}^{(V,r)}$, and $\widehat{W}^{(P,r)}$ denote the question for the r'th verifier player and r'th prover player respectively. The question $\widehat{W}^{(V,r)}$ is a 6-tuple $(\widehat{W}_1^{(V,r)}, \ldots, \widehat{W}_6^{(V,r)})$ of commuting two-qubit Pauli observables.
- For $r \in [N_V]$, for $j \in \{1, \ldots, 6\}$, we overload notation by also letting $\widehat{W}_j^{(V,r)}$ denote the j'th Pauli observable used by r'th verifier player in the honest strategy $\mathcal{S}_{ZK}$ when they receive question $\widehat{W}^{(V,r)}$, as specified in Section 3.2.1. We also let $\widehat{W}^{(V,r)}$ denote the observable that is the product $\widehat{W}_1^{(V,r)} \cdots \widehat{W}_6^{(V,r)}$ (the order does not matter because the observables commute). Whether or not $\widehat{W}_j^{(V,r)}$ and $\widehat{W}^{(V,r)}$ are used to refer to the question or the observables will be clear from context.
- For $r \in [N_P]$, we let $\widehat{W}^{(P,r)}$ also denote the observable used by prover player PP_r in the honest strategy $\mathcal{S}_{ZK}$ when they receive question $\widehat{W}^{(P,r)}$. For example, if $\widehat{W}^{(P,r)}$ is a "Question gate" Q_{rj} or a "Question flag flip" QF_r, then as an observable we interpret $\widehat{W}^{(P,r)}$ as the corresponding Pauli observable in the honest strategy $\mathcal{S}_{ZK}$. If $\widehat{W}^{(P,r)}$ is a prover reflection $\star_r$, then as an observable we interpret $\widehat{W}^{(P,r)}$ as P'_r.
- Let $\widehat{W} = \left(\widehat{W}^{(D,r)}\right)_{D \in \{V,P\}, r \in [N_D]}$ denote the tuple of questions for all players in the protocol. We also use $\widehat{W}$ to denote the tensor product of observables
$$\widehat{W} = \bigotimes_{D \in \{V,P\}, r \in [N_D]} \widehat{W}^{(D,r)}.$$
- For $r \in [N_V]$, $j \in \{1, \ldots, 6\}$, we define the observable $\widetilde{W}_j^{(V,r)}$ to be $\widehat{W}_j^{(V,r)}$. For $r \in [N_P]$, we define the observable
$$\widetilde{W}^{(P,r)} = \begin{cases} \sigma_X(\mathsf{F}_{P_r}) & \text{if } \widehat{W}^{(P,r)} = \star_r \\ \widehat{W}^{(P,r)} & \text{otherwise} \end{cases}$$

  Notice that the observables $\widetilde{W}^{(D,r)}$ are simply Pauli observables (or products of Pauli observables). We explain the reasoning behind defining the observables $\widetilde{W}^{(D,r)}$ in the next section.

- We also define the projectors corresponding to the players' observables. The verifier players output a 6-tuple of bits $(b_1, \dots, b_6) \in \{0,1\}^6$. For $r \in [N_V]$, $j \in \{1, \dots, 6\}$, and bit $b \in \{0,1\}$, define
$$\widehat{W}_j^{(V,r)}(b) = \frac{1}{2}\left(I + (-1)^b \widehat{W}_j^{(V,r)}\right)$$
which is the projector onto the b subspace of $\widehat{W}_j^{(V,r)}$. Define
$$\widehat{W}^{(V,r)}(b_1, \dots, b_6) = \prod_{i=1}^{6} \widehat{W}_i^{(V,r)}(b_i).$$
The prover players only output a single bit, so for $r \in [N_P]$, define
$$\widehat{W}^{(P,r)}(b) = \frac{1}{2}\left(I + (-1)^b \widehat{W}^{(P,r)}\right).$$
Let $a = \left(a^{(D,r)}\right)_{D \in \{V,P\}, r \in [N_D]}$ denote an answer vector for all players (where $a^{(V,r)}$ corresponds to a 6-tuple of bits). Then for every tuple of questions $\widehat{W}$, we define
$$\widehat{W}(a) = \bigotimes_{D \in \{V,P\}, r \in [N_D]} \widehat{W}^{(D,r)}(a^{(D,r)}).$$
We define the projectors $\widetilde{W}^{(D,r)}(b)$ and $\widetilde{W}^{(D,r)}$ analogously.

For convenience we also provide a notation reference table in Figure 2.

4.1 Non-adaptive cheating referees

In this section, we show that for *every* possible combination of (correctly formatted) questions to the players, the joint distribution of answers of players using the honest strategy $\mathcal{S}_{ZK}$ is efficiently simulable.

The reason for defining the observables $\widetilde{W}^{(D,r)}$ is as follows. Ultimately, the goal of the simulator is, for every question tuple $\widehat{W}$, to sample answer vectors a that is distributed according to the probability density
$$\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi)\, \widehat{W}(a)\right)$$
where $\Phi = |\Phi\rangle\langle\Phi|$ is the shared entangled state and $\widehat{W}(a)$ also denotes the projectors corresponding to outcome a in the honest strategy $\mathcal{S}_{ZK}(x)$ (see Section 3.2.1). The main difficulty is that the simulator does not have any control over the prover reflections, nor the parts of Φ that correspond to the provers' private registers (which may be unbounded in size).

To get around this issue, the key observation we use is the following: the measurements of the verifier players are Pauli observables that act on at most a constant number of qubits. Furthermore, the measurements of the prover players when they get a question other than the special prover reflection $\star$ are also just Pauli observables on a constant number of qubits.

We define two notions of support of a question tuple $\widehat{W}$. Then, define the *physical support* of $\widehat{W}$ to be the set $S'_{\widehat{W}}$ of qubit registers that are acted upon nontrivially by $\widehat{W}$, omitting the prover players' private P_r registers. The set $S'_{\widehat{W}}$ contains subregisters of $\mathsf{C}', \mathsf{V}', \mathsf{M}, \mathsf{F}$.

We can also define the *logical support* of $\widehat{W}$, denoted by the set $S_{\widehat{W}}$, which contains subregisters of $\mathsf{C}, \mathsf{V}, \mathsf{M}, \mathsf{F}$ that correspond to the registers in $S'_{\widehat{W}}$. The logical support set $S_{\widehat{W}}$ contains all the M_{ij}

and F_i registers that are in $S'_{\widehat{W}}$. The set $S_{\widehat{W}}$ contains C_t if and only if $S'_{\widehat{W}}$ contains C_{tj} for some j, and similarly contains V_i if and only if $S'_{\widehat{W}}$ contains V_{ij} for some j. The difference between the physical and logical support of $\widehat{W}$ comes from the fact that the history state $|\Phi\rangle$ was encoded using $\mathcal{C}_{outer}$ and split between multiple provers.

Note that the number of qubit registers in $S_{\widehat{W}}$ is at most $12N_V + k$. This is because the questions to each verifier player is a 6-tuple of Pauli observables that act on up to 2 qubits, and each prover player measures at most a single qubit flag register at a time. Define $L = 12N_V = 48$, which is the maximum number of verifier player qubits that can be addressed by $\widehat{W}$.

For all $\widehat{W}$, the simulator computes a *succinct* description of density matrix ρ defined only on the logical registers in $S_{\widehat{W}}$ that mimics Φ in a certain sense that is captured by the following Lemma 15. Before stating the Lemma, however, we specify what we mean by succinct description of ρ. In general, ρ will be a density matrix with dimension at least 2^{12N_V+k}, so the naïve strategy of explicitly storing all the matrix entires of ρ is not an efficient representation if the number of prover players k is a growing function. Instead, we will specify our density matrices ρ and measurement operators using the following type of efficient representation:

Definition 13 (Efficient representations of operators)**.** *Let A denote a linear operator defined on m qubits. The operator A has an (w,ℓ)-*efficient representation *if there exist, for all $i \in \{1,2,\dots,w\}$, a collection of operators $\{A_{ij}\}$ where each A_{ij} is defined on some subset $S_{ij} \subseteq \{1,2,\dots,m\}$ of qubit registers, and*

1. *For all $i \in \{1,2,\dots,w\}$, $\{S_{ij}\}_j$ is a partition of $\{1,2,\dots,m\}$.*
2. *$|S_{ij}| \le \ell$ for all i,j.*
3. *The explicit matrix representation of A_{ij} can be described using $2^{O(\ell)}$ bits.*
4. *$A = \sum_{t=1}^{w} \bigotimes_j A_{ij}$.*

The following Claim justifies our definition of "efficient representation":

Claim 14. *Let A, B be m-qubit operators with (w,ℓ)-efficient representations $\{A_{ij}\}$ and $\{B_{ij}\}$, respectively. First, the efficient representations of both operators have bit complexity $w \cdot 2^{O(\ell)} \cdot \mathrm{poly}(m)$. Second, the trace $\mathrm{Tr}(AB)$ can be computed in time $w \cdot 2^{O(\ell)} \cdot \mathrm{poly}(m)$.*

We can now state our main simulation Lemma:

Lemma 15. *There is a PPT algorithm* SimDensity *that when given a tuple $\widehat{W}$ of questions, outputs a $(3(T+1)L^2, 4L)$-efficient representation of a density matrix ρ such that for all answer vectors $a = \left(a^{(D,r)}\right)_{D\in\{V,P\}, r\in[N_D]}$, we have that*

$$\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi)\,\widehat{W}(a)\right) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\rho)\,\widetilde{W}(a)\right). \tag{3}$$

Furthermore, the density matrix ρ is defined on the logical support $S_{\widehat{W}}$ of $\widehat{W}$.

Before proving Lemma 15, we first prove a specialized version. Let $1 \le t_1 \le t_2 \le T$ be such that $t_2 - t_1 \le L$ and let $I(t_1,t_2) = \{t : t_1 \le t \le t_2\}$ denote the interval of time steps between t_1 and t_2. We show that we can simulate measurements on the state $\mathsf{Enc}_{outer}(\Phi_I)$ where $|\Phi_{I(t_1,t_2)}\rangle$ is the post-measurement state

$$|\Phi_{I(t_1,t_2)}\rangle = \frac{1}{\sqrt{t_2 - t_1 + 1}} \sum_{t \in I(t_1,t_2)} |\mathsf{unary}(t)\rangle_{\mathsf{C}} \otimes |\Phi_t\rangle_{\mathsf{VMPF}}$$

In other words, $|\Phi_{I(t_1,t_2)}\rangle$ denotes the part of the history state between times t_1 and t_2. Furthermore, when convenient we will omit mention of the unary encoding of the clock, and simply refer to the state of the clock register as $|t\rangle$.

Lemma 16. *There is a PPT algorithm* SimInterval *that when given a tuple $\widehat{W}$ of questions and a pair of times $0 \le t_1 \le t_2 \le T$ such that $t_2 - t_1 \le L$, outputs a $(3L^2, 4L)$-efficient representation of a density matrix ρ such that for all answer vectors $a = \left(a^{(D,r)}\right)_{D \in \{V,P\}, r \in [N_D]}$, we have that*

$$\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_{I(t_1,t_2)})\,\widehat{W}(a)\right) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\rho)\,\widetilde{W}(a)\right). \tag{4}$$

Furthermore, the density matrix ρ is defined on the logical support $S_{\widehat{W}}$ of $\widehat{W}$.

Proof. Let $I = I(t_1,t_2)$ and $S = S_{\widehat{W}}$. Because of padding, we can assume without loss of generality that the time interval I belongs entirely to one of the six phases of the protocol circuit V_{enc} defined in Section 3.1.

We can write for all $t \in I$,

$$|\Phi_t\rangle = |\Delta_t\rangle_{\mathsf{VMP}} \otimes |f(t)\rangle_{\mathsf{F}}. \tag{5}$$

Thus, we have

$$|\Phi_I\rangle\langle\Phi_I| = \frac{1}{|I|}\sum_{t,t' \in I} |t\rangle\langle t'|_{\mathsf{C}} \otimes |\Delta_t\rangle\langle\Delta_{t'}|_{\mathsf{VMP}} \otimes |f(t)\rangle\langle f(t')|_{\mathsf{F}}.$$

The left hand side of (4) can be written as

$$\begin{aligned}&\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_I)\,\widehat{W}(a)\right)\\ &= \frac{1}{|I|}\sum_{t,t'\in I}\mathrm{Tr}\left(\mathsf{Enc}_{outer}\left(|t\rangle\langle t'| \otimes |\Delta_t\rangle\langle\Delta_{t'}|\right) \otimes |f(t)\rangle\langle f(t')|\,\widehat{W}(a)\right)\end{aligned} \tag{6}$$

We consider two cases.

Case 1. First, suppose that the following holds for all $r \in [N_P]$: either the r'th prover flag $p_r(t)$ stays constant throughout the interval I, or if it changes, then $\widehat{W}^{(P,r)} \neq \star$ (that is, prover player PP_r was not asked a $\star$ question).

Fix a $t, t' \in I$. Let $\widehat{W}^\star(a)$ denote the tensor factors of $\widehat{W}(a)$ corresponding to the prover players who received a $\star$ question (if none received a $\star$ question, then this operator is the identity). Similarly, let $\widetilde{W}^\star(a)$ denote the tensor factors of $\widetilde{W}(a)$ corresponding to the prover players who received a $\star$ question. Thus, $\widetilde{W}^\star(a)$ is tensor product of σ_X operators and identity operators. Under our assumption, any operator A defined on registers CVMP, we have that

$$\mathrm{Tr}\left(\left(A \otimes |f(t)\rangle\langle f(t')|_{\mathsf{F}}\right)\widehat{W}^\star(a)\right) = \mathrm{Tr}\left(\left(A \otimes |f(t)\rangle\langle f(t')|_{\mathsf{F}}\right)\widetilde{W}^\star(a)\right).$$

This is because of the following: consider the set $J \subseteq [N_P]$ who received a $\star$ question. If J is empty, then $\widehat{W}^\star = \widetilde{W}^\star = \mathbb{I}$,[9] so the equation trivially holds. If J is non-empty, then by assumption for any $r \in J$, the prover flags for r stay constant on the interval I, so the traces are 0. This implies that (6) is equal to

$$\frac{1}{|I|}\sum_{t,t'\in I}\mathrm{Tr}\left(\mathsf{Enc}_{outer}\left(|t\rangle\langle t'| \otimes |\Delta_t\rangle\langle\Delta_{t'}|\right) \otimes |f(t)\rangle\langle f(t')|\,\widetilde{W}(a)\right)$$

[9] We denote the identity matrix as $\mathbb{I}$ here in order to avoid confusion with the interval I.

We now argue that a $(1, 4L)$-efficient representation of the following operator

$$\rho_S(t,t') = \mathrm{Tr}_{\overline{S}}\left(|t\rangle\langle t'| \otimes |\Delta_t\rangle\langle\Delta_{t'}| \otimes |f(t)\rangle\langle f(t')|\right)$$

can be efficiently computed in polynomial time, where $\mathrm{Tr}_{\overline{S}}(\cdot)$ denotes tracing out all registers except those in S. Notice that S does not include the register P, and has at most $O(N_V + k)$ qubit registers.

Given this is true, and using the fact that $|I| \leq L$, then a $(L^2, 4L)$-efficient representation of

$$\rho_S = \frac{1}{|I|}\sum_{t,t' \in I} \rho_S(t,t')$$

can be computed in polynomial time, and satisfies (4).

Since $\mathrm{Tr}_{\overline{S}}\left(|t\rangle\langle t'|\right)$ and $\mathrm{Tr}_{\overline{S}}\left(|f(t)\rangle\langle f(t')|\right)$ have $(1,1)$-efficient representations (i.e. these are tensor products of single qubit operators), it suffices to show that we can efficiently compute $\mathrm{Tr}_{\overline{S}}(|\Delta_t\rangle\langle\Delta_{t'}|)$. Assume without loss of generality that $t \leq t'$.

First, we consider the sub-case that all prover flags $\{p_r(t)\}$ stay constant throughout the interval I. This means that there exists a sequence of elementary gates $g_t, g_{t+1}, \ldots, g_{t'}$ (i.e., no prover gates) such that

$$|\Delta_{t'}\rangle = g_{t'}g_{t'-1}\cdots g_{t+1}g_t|\Delta_t\rangle.$$

Let G denote the union of the registers that are acted upon by the gates $g_t, \ldots, g_{t'}$. Since $t' - t \leq |I| \leq L$, and each gate acts on at most 3 qubits, we get that $|G| \leq 3L$. Now, we can write

$$\begin{aligned}\mathrm{Tr}_{\overline{S}}(|\Delta_t\rangle\langle\Delta_{t'}|) &= \mathrm{Tr}_{\overline{S}}(|\Delta_t\rangle\langle\Delta_t| g_{t'}^\dagger \cdots g_t^\dagger) \\ &= \mathrm{Tr}_{G\cap\overline{S}}\left(\mathrm{Tr}_{\overline{S\cup G}}(|\Delta_t\rangle\langle\Delta_t|) g_{t'}^\dagger \cdots g_t^\dagger\right)\end{aligned}$$

The density matrix $\mathrm{Tr}_{\overline{S\cup G}}(|\Delta_t\rangle\langle\Delta_t|)$ is where all registers except for S and G are traced out. We notice that the number of qubits of this density matrix, $|S \cup G|$, is at most $4L$. We can appeal to the following Lemma to get that the explicit matrix description of $\mathrm{Tr}_{\overline{S\cup G}}(|\Delta_t\rangle\langle\Delta_t|)$ can be computed in polynomial time.

Lemma 17. *There exists a PPT algorithm* $\mathrm{SimSnapshot}$ *that on input* (x, Y, t) *such that*

1. *x is a binary string*
2. *Y is a subset of registers used in the honest strategy $\mathcal{S}_{ZK}(x)$ (that does not include the prover registers P nor the prover flags F) that has size at most $4L$, and*
3. *t is an integer between 0 and the length of the protocol circuit $V_{enc}(x)$*

outputs matrix entries of the density matrix

$$\mathrm{Tr}_{\overline{Y}}\left(|\Delta_t\rangle\langle\Delta_t|\right)$$

where $|\Delta_t\rangle$ is defined as in (5).

We defer the proof of Lemma 17 to Section 4.3.

The simulator $\mathrm{SimInterval}$ can execute $\mathrm{SimSnapshot}$ on $(x, S \cup G, t)$ to obtain the description of $\mathrm{Tr}_{\overline{S\cup G}}(|\Delta_t\rangle\langle\Delta_t|)$, and then perform some efficient post-processing to obtain the explicit matrix description of $\mathrm{Tr}_{\overline{S}}(|\Delta_t\rangle\langle\Delta_{t'}|)$.

Putting everything together, we get that

$$\rho_S(t,t') = \mathrm{Tr}_{\overline{S}}\left(|t\rangle\langle t'|\right) \otimes \mathrm{Tr}_{\overline{S}}\left(|\Delta_t\rangle\langle\Delta_{t'}|\right) \otimes \mathrm{Tr}_{\overline{S}}\left(|f(t)\rangle\langle f(t')|\right)$$

has a $(1, 4L)$-efficient representation.

Next, we consider the next sub-case, where the prover flags $\{p_r(t)\}$ do not stay constant. The interval I lies within the Prover Operation phase. Because of padding, the interval I can at most cover a single prover's operation, so there exists a unique $r^* \in [N_P]$ such that $p_{r^*}(t)$ changes (all others stay constant). Thus, the p_{r^*} flag changes from 0 to 1 at time $t_\star(r^*)$. Let $t_\star = t_\star(r^*)$.

By our assumption at the beginning, $\widehat{W}^{(P,r^*)} \neq \star$ (the prover player PP_{r^*} was not asked the $\star$ question). Since S does not include $\mathsf{F}_{P_{r^*}}$, for $t < t_\star$ and $t' \geq t_\star$, we get that

$$\mathrm{Tr}_{\overline{S}}\left(|f(t)\rangle\langle f(t')|\right) = 0,$$

Define $I^- = \{t \in I : t < t_\star\}$ and $I^+ = \{t \in I : t \geq t_\star\}$. We then have

$$\rho_S = \frac{1}{|I|}\left(\sum_{t,t' \in I^-} \rho_S(t,t') + \sum_{t,t' \in I^+} \rho_S(t,t')\right).$$

Notice that all prover flags $p_r(t)$ stay constant on I^- and I^+. Therefore we can reduce to the previous sub-case to argue that $\rho_S(t,t')$ can be computed when both t, t' either come from I^- or I^+.

This completes the proof of Case 1.

Case 2. Next, we consider the case that there is an r^* for which the prover flag $p_{r^*}(t)$ changes from 0 to 1 during the interval I, and furthermore $\widehat{W}^{(P,r^*)} = \star$ (the prover player PP_{r^*} was asked the $\star$ question). Again, this interval I must lie in the Prover Operation phase and by padding all other prover flags must be constant throughout the interval I. Let $t_\star = t_\star(r^*)$.

Since prover player PP_{r^*} received the $\star$ question, they could not have received questions QF_{r^*} or AF_{r^*}, and therefore $\mathsf{F}_{Q_{r^*}}$ and $\mathsf{F}_{A_{r^*}}$ are not part of the logical support set S. Thus the reduced density matrix of Φ_I where we trace out all registers except for S and P is a convex combination

$$\mathrm{Tr}_{\overline{S \cup \mathsf{P}}}(\Phi_I) = \mathrm{Tr}_{\overline{S \cup \mathsf{P}}}\left(\frac{|I^-|}{|I|}\Phi_{I^-} + \frac{|I^\star|}{|I|}\Phi_{I^\star} + \frac{|I^+|}{|I|}\Phi_{I^+}\right)$$

where we define the subintervals

- $I^- = \{t \in I : t < t_\star - 1\}$
- $I^\star = \{t \in I : t_\star - 1 \leq t \leq t_\star\}$
- $I^+ = \{t \in I : t \geq t_\star + 1\}$

This is because we are tracing out the Question Flip flag register $\mathsf{F}_{Q_{r^*}}$ and Answer Flip flag register $\mathsf{F}_{A_{r^*}}$; so cross-terms where t and t' belong to different subintervals above would disappear.

Therefore (6) is equal to

$$\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_I)\,\widehat{W}(a)\right) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}\left(\frac{|I^-|}{|I|}\Phi_{I^-} + \frac{|I^\star|}{|I|}\Phi_{I^\star} + \frac{|I^+|}{|I|}\Phi_{I^+}\right)\widehat{W}(a)\right)$$

We now show how to compute $(L^2, 4L)$-efficient representations of density matrices $\rho_S^-, \rho_S^+, \rho_S^\star$ such that

$$\mathrm{Tr}(\mathsf{Enc}_{outer}(\rho_S^-)\,\widetilde{W}(a)) = \mathrm{Tr}(\mathsf{Enc}_{outer}(\Phi_{I^-})\,\widehat{W}(a)) \quad (7)$$

$$\mathrm{Tr}(\mathsf{Enc}_{outer}(\rho_S^+)\,\widetilde{W}(a)) = \mathrm{Tr}(\mathsf{Enc}_{outer}(\Phi_{I^+})\,\widehat{W}(a)) \quad (8)$$

$$\mathrm{Tr}(\mathsf{Enc}_{outer}(\rho_S^\star)\,\widetilde{W}(a)) = \mathrm{Tr}(\mathsf{Enc}_{outer}(\Phi_{I^\star})\,\widehat{W}(a)). \quad (9)$$

Once we have this, then a $(3L^2, 4L)$-efficient representation of density matrix $\rho_S = \frac{|I^-|}{|I|}\rho_S^- + \frac{|I^\star|}{|I|}\rho_S^\star + \frac{|I^+|}{|I|}\rho_S^+$ is efficiently computable and satisfies (4), and this completes the proof of Case 2.

We argue that ρ_S^- and ρ_S^+ have efficient representations. Notice that the prover flags $\{p_r(t)\}$ are constant on the intervals I^- and I^+. Thus from the same arguments as in Case 1, $\mathrm{SimInterval}$ can, when given input $\widehat{W}$ and a pair of times $(\min(I^-), \max(I^-))$, efficiently compute a $(L^2, 4L)$-efficient representation of the density matrix ρ_S^- defined on S that satisfies (7). Similarly, $\mathrm{SimInterval}$ can also efficiently compute an efficient representation of ρ_S^+ that satisfies (8).

We now turn to $\rho_S^\star$. Since we are in Case 2, it must be that $I^\star = \{t_\star - 1, t_\star\}$ (otherwise, the prover flag for PP_{r^*} would stay constant on I). Thus, using that $|\Phi_t\rangle = |\Delta_t\rangle \otimes |f(t)\rangle$,

$$\begin{aligned}|\Phi_{I^\star}\rangle &= \frac{1}{\sqrt{2}}\Big[|t_\star - 1\rangle|\Delta_{t_\star-1}\rangle|f(t_\star-1)\rangle + |t_\star\rangle|\Delta_{t_\star}\rangle|f(t_\star)\rangle\Big] \\ &= \frac{1}{\sqrt{2}}\Big[|t_\star - 1\rangle \otimes |\Phi_{t_\star-1}\rangle + |t_\star\rangle \otimes P'_{r^*}|\Phi_{t_\star-1}\rangle\Big] \qquad (10)\end{aligned}$$

Furthermore, since PP_{r^*} receives the $\star$ question in $\widehat{W}$, the measurement operator $\widehat{W}^{(P,r^*)}(a)$ is simply

$$\frac{1}{2}\left(\mathbb{I} + (-1)^{a^{(P,r^*)}} P'_{r^*}\right).$$

Let $\widehat{W}^{-(P,r*)}(a)$ be the measurement operator obtained by taking $\widehat{W}(a)$ and deleting the factor $\widehat{W}^{(P,r^*)}$, i.e., $\widehat{W}^{-(P,r*)}(a)$ is the tensor product of questions of all provers except PP_{r^*}. Therefore we can write

$$\begin{aligned}&\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_{I^\star})\,\widehat{W}(a)\right) \\ &= \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_{I^\star})\,\widehat{W}^{-(P,r^*)}(a) \otimes \widehat{W}^{(P,r^*)}(a)\right) \\ &= \frac{1}{2}\left(\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_{I^\star})\,\widehat{W}^{-(P,r^*)}(a)\right) + (-1)^{a^{(P,r^*)}}\,\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_{I^\star})\,\widehat{W}^{-(P,r^*)}(a) \otimes P'_{r^*}\right)\right) \qquad (11)\end{aligned}$$

We analyze the first term above. By substituting in the expression (10) for $\Phi_{I^\star}$, we get some cross terms of the form

$$\begin{aligned}&\mathrm{Tr}\left(\mathsf{Enc}_{outer}(|t_\star - 1\rangle\langle t_\star| \otimes |\Phi_{t_\star-1}\rangle\langle\Phi_{t_\star-1}|(P'_{r^*})^\dagger)\,\widehat{W}^{-(P,r^*)}(a)\right) \\ &= \mathrm{Tr}\left(\Big[\mathsf{Enc}_{outer}(|t_\star - 1\rangle\langle t_\star| \otimes |\Delta_{t_\star-1}\rangle\langle\Delta_{t_\star-1}|P^\dagger_{r^*}) \otimes |f(t_\star-1)\rangle\langle f(t_\star)|\Big]\,\widehat{W}^{-(P,r^*)}(a)\right)\end{aligned}$$

Notice that the operator $\widehat{W}^{-(P,r^*)}(a)$ does not act on the prover flag register $\mathsf{F}_{P_{r^*}}$, and the $\mathsf{F}_{P_{r^*}}$ component of $|f(t_\star - 1)\rangle\langle f(t_\star)|$ is $|0\rangle\langle 1|$. Thus, the cross-term vanishes. The first term of (11) can be written as

$$\begin{aligned}&\frac{1}{2}\mathrm{Tr}\left(\mathsf{Enc}_{outer}(|t_\star - 1\rangle\langle t_\star - 1| \otimes |\Phi_{t_\star-1}\rangle\langle\Phi_{t_\star-1}| + |t_\star\rangle\langle t_\star| \otimes |\Phi_{t_\star}\rangle\langle\Phi_{t_\star}|)\,\widehat{W}^{-(P,r^*)}(a)\right) \\ &= \frac{1}{2}\mathrm{Tr}\left(\mathsf{Enc}_{outer}\left((|t_\star - 1\rangle\langle t_\star - 1| + |t_\star\rangle\langle t_\star|) \otimes |\Phi_{t_\star-1}\rangle\langle\Phi_{t_\star-1}|\right)\,\widehat{W}^{-(P,r^*)}(a)\right)\end{aligned}$$

where in the equality we used that $|\Phi_{t_\star}\rangle = P'_{r^*}|\Phi_{t_\star-1}\rangle$ and the operator P'_{r^*} commutes with $\widehat{W}^{-(P,r^*)}(a)$, and thus vanishes by the cyclity of the trace. Applying similar reasoning to the second term of (10), we remain only with the cross terms, and we get that it can be written as

$$\frac{1}{2}\mathrm{Tr}\left(\mathsf{Enc}_{outer}\left((|t_\star - 1\rangle\langle t_\star| + |t_\star\rangle\langle t_\star - 1|) \otimes |\Phi_{t_\star-1}\rangle\langle\Phi_{t_\star-1}|\right)\,\widehat{W}^{-(P,r^*)}(a)\right)$$

Putting everything together, we get that (11) can be written as

$$\frac{1}{2}\,\mathrm{Tr}\left(\mathsf{Enc}_{outer}\left(\tau(a,t_\star)\otimes|\Delta_{t_\star-1}\rangle\langle\Delta_{t_\star-1}|\otimes|f(t_\star-1)\rangle\langle f(t_\star-1)|\right)\widehat{W}^{-(P,r^*)}(a)\right) \tag{12}$$

with

$$\tau(a,t_\star)=\frac{1}{2}\Big[|t_\star-1\rangle+(-1)^{a^{(P,r^*)}}|t_\star\rangle\Big]\Big[\langle t_\star-1|+(-1)^{a^{(P,r^*)}}\langle t_\star|\Big].$$

Define

$$\rho^\star_S=\mathrm{Tr}_{\overline{S}}\left(\tau(a,t_\star)\otimes|\Delta_{t_\star-1}\rangle\langle\Delta_{t_\star-1}|\otimes|f(t_\star-1)\rangle\langle f(t_\star-1)|\right).$$

Just like in Case 1, the density matrices $\mathrm{Tr}_{\overline{S}}\left(\tau(a,t_\star)\right)$ and $\mathrm{Tr}_{\overline{S}}\left(|f(t_\star-1)\rangle\langle f(t_\star-1)|\right)$ have $(1,1)$-efficient representations, and by Lemma 17 we have that $\mathrm{Tr}_{\overline{S}}\left(|\Delta_{t_\star-1}\rangle\langle\Delta_{t_\star-1}|\right)$ can be efficiently computed as well. This shows that $\rho^\star_S$ has a $(1,4L)$-efficient representation. Finally, we have that the $\widehat{W}^{-(P,r^*)}(a)$ operator in (12) can be replaced with $\widetilde{W}^{-(P,r^*)}(a)$. This shows that $\rho^\star_S$ satisfies (9), and this completes the proof of Case 2.

□

We now prove Lemma 15.

Proof of Lemma 15. Fix a tuple $\widehat{W}$ of questions. We argue that computing an efficient description of a density matrix ρ that satisfies (3) can be efficiently reduced to computing efficient descriptions of density matrices ρ_I for various intervals I, for which we can use the algorithm $\mathrm{SimInterval}$ from Lemma 16.

Since there are only N_V verifier players, and each verifier player receives a 6-tuple of Pauli observables that have support on at most 12 physical qubits each, the joint measurement of the verifier players acts on at most $12N_V$ physical qubits, and therefore at most $12N_V$ logical qubits of the underlying encoded clock register.

Let

$$C_{tr}=\{i\in[T]:\text{the }i\text{'th logical clock qubit }\mathsf{C}_i\text{ is not in }S_{\widehat{W}}\}$$

denote the set of (logical) clock qubit registers that, after the outer encoding, are not acted upon by the measurement corresponding to $\widehat{W}$. Thus, for all answer vectors a,

$$\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi)\,\widehat{W}(a)\right)=\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\mathrm{Tr}_{C_{tr}}(\Phi))\,\widehat{W}(a)\right). \tag{13}$$

We argue that the density matrix $\mathrm{Tr}_{C_{tr}}(\Phi)$ is a convex combination of $|\Phi_I\rangle$ states for various intervals I:

$$\mathrm{Tr}_{C_{tr}}(\Phi)=\frac{1}{T+1}\sum_{t,t'}\mathrm{Tr}_{C_{tr}}(|\mathsf{unary}(t)\rangle\langle\mathsf{unary}(t')|)\otimes|\Phi_t\rangle\langle\Phi_{t'}|. \tag{14}$$

The following Claim easily follows from the structure of unary encodings:

Claim 18. *For all $0\le t,t'\le T$, the operator $\mathrm{Tr}_{C_{tr}}(|\mathsf{unary}(t)\rangle\langle\mathsf{unary}(t')|)$ is non-zero only when $t=t'$, or for all $i\in C_{tr}$, either both $t,t'>i$, or both $t,t'<i$.*

Given this Claim, we notice that all cross-terms of (14) involving times t,t' where $t\ne t'$ and at least one of t,t' are in C_{tr} vanish. Thus the only cross-terms that remain are times t,t' that come from an interval $I\subseteq\{0,1,2,\ldots,T\}$ of consecutive time-steps where there is no $i\in C_{tr}$ such that

$\min(I) \leq i \leq \max(I)$. Let $\{0, 1, 2, \ldots, T\} \setminus C_{tr}$ be the union of maximal intervals $I_1, I_2, \ldots, I_\ell$ of consecutive time steps. Thus (14) can be written as

$$\sum_{t \in C_{tr}} \frac{1}{T+1} \mathrm{Tr}_{C_{tr}}(|\Phi_{\{t\}}\rangle\langle\Phi_{\{t\}}|) + \sum_{j=1}^{\ell} \frac{|I_j|}{T+1} \mathrm{Tr}_{C_{tr}}(|\Phi_{I_j}\rangle\langle\Phi_{I_j}|)$$

where $|\Phi_{\{t\}}\rangle = |\mathsf{unary}(t)\rangle \otimes |\Phi_t\rangle$ denotes the history state restricted to the singleton interval $\{t\}$. As desired, (14) is a probabilistic mixture of interval states $|\Phi_I\rangle$ where each interval has size at most $6N_V \leq L$. The intervals I_j occur with probability $|I_j|/(T+1)$ and the singleton intervals $\{t\}$ for $t \in C_{tr}$ occur with probability $1/(T+1)$.

The algorithm $\mathrm{SimDensity}$ works as follows: given a question tuple $\widehat{W}$ it can compute the set C_{tr}, and then compute the intervals $I_1, \ldots, I_j$ in polynomial time. For each interval I_j, it invokes the algorithm $\mathrm{SimInterval}$ from Lemma 16 to efficiently compute a $(3L^2, 4L)$-efficient representation of the density matrix ρ_{I_j} supported on $S_{\widehat{W}}$ that satisfies

$$\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_{I_j})\, \widehat{W}(a)\right) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\rho_{I_j})\, \widetilde{W}(a)\right).$$

Similarly, for every $t \in C_{tr}$ the algorithm $\mathrm{SimDensity}$ invokes $\mathrm{SimInterval}$ to compute a $(3L^2, 4L)$-efficient representation of the density matrix ρ_t that satisfies

$$\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi_{\{t\}})\, \widehat{W}(a)\right) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\rho_t)\, \widetilde{W}(a)\right).$$

There are at most $T+1$ density matrices to compute. $\mathrm{SimDensity}$ then can then efficiently compute a $(3(T+1)L^2, 4L)$-efficient representation of the convex combination

$$\rho = \sum_{t \in C_{tr}} \frac{1}{T+1} \rho_t + \sum_{j=1}^{\ell} \frac{|I_j|}{T+1} \rho_{I_j},$$

which satisfies (3).

□

With Lemma 15, we prove that V_{ZK} has the zero knowledge property against cheating referees that are non-adaptive, meaning that the referee samples a question tuple $\widehat{W}$ first, sends them to the players, and receives their answers a.

Lemma 19. *For every non-adaptive polynomial-time referee $\widehat{R}^{na}$, there is a PPT simulator $\mathrm{Sim}_{\widehat{R}^{na}}$ such that the output distribution of $\mathrm{Sim}_{\widehat{R}^{na}}(x)$ is equal to $\mathrm{View}(\widehat{R}^{na}(x) \leftrightarrow \mathcal{S}_{ZK}(x))$.*

Proof. $\mathrm{Sim}_{\widehat{R}^{na}}$ starts by sampling the questions to the players $\widehat{W} = \left(\widehat{W}^{(D,r)}\right)_{D \in \{V,P\}, r \in [N_D]}$ from the same joint distribution as $\widehat{R}^{na}$ on input x. This can be performed efficiently since $\widehat{R}^{na}$ is a polynomial-time algorithm and the questions are sampled in a non-adaptive way.

Then, the simulator $\mathrm{Sim}_{\widehat{R}^{na}}$ executes the algorithm $\mathrm{SimDensity}$ from Lemma 15 on input $\widehat{W}$, which outputs an efficient representation of a density matrix ρ such that for all answer vectors $a = \left(a^{(D,r)}\right)_{D \in \{V,P\}, r \in [N_D]}$ we have that

$$\alpha(a) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi)\, \widehat{W}(a)\right) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\rho)\, \widetilde{W}(a)\right).$$

Note that $\alpha(a)$ is a probability distribution over answer vectors. We need to show that we can efficiently sample an answer vector a from the probability distribution $\alpha(a)$. We can do that by sampling each bit of a one at a time, and conditioning the density matrix ρ on the partial outcomes.

Index the players using $\{1, 2, \ldots, N_V + k\}$ in some canonical way. Let $a = (a_1, \ldots, a_{N_V+k})$ where a_i denotes the answer symbol of the i'th player, which might come from the alphabet $\{0, 1, \}^6$ or $\{0, 1\}$, depending on whether the i'th player is a prover player or a verifier player.

We utilize the following important observation: for every answer vector a, $\widetilde{W}(a)$ is equal to the tensor product of projectors where the projectors corresponding to the prover players are all single-qubit operators, and the projectors corresponding to the verifier players may act on up to $12N_V$ qubits.

For every $i \in \{1, 2, \ldots, N_V + k\}$, let $\widetilde{W}(a_i)$ denote the projector of the i'th player corresponding to outcome a_i, when the players receive the question tuple $\widehat{W}$. Note that $\widetilde{W}(a) = \widetilde{W}(a_1) \otimes \cdots \otimes \widetilde{W}(a_{N_V+k})$.

To sample a_1, the simulator can explicitly compute the probabilities

$$\alpha(a_1) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\Phi)\,\widehat{W}(a_1)\right) = \mathrm{Tr}\left(\mathsf{Enc}_{outer}(\rho)\,\widetilde{W}(a_1)\right).$$

for all a_1, where we use $\alpha(a_1)$ to denote the marginal distribution of a_1 in α. Since a_1 comes from a constant-sized alphabet, this distribution can be sampled from in polynomial time. Given a sample a_1, we can now sample a_2 *conditioned* on a_1, so we can compute the conditional distribution

$$\alpha(a_2|a_1) = \frac{\mathrm{Tr}\left(\mathsf{Enc}_{outer}(\rho)\,\widetilde{W}(a_1) \otimes \widetilde{W}(a_2)\right)}{\alpha(a_1)},$$

and sample from it as well. We can continue in this manner, until we have sampled $a_1 \cdots a_{N_V+k}$. This can be done in polynomial time, because $\widetilde{W}(a_1) \otimes \cdots \otimes \widetilde{W}(a_i)$ for all $i \in \{1, \ldots, N_V + k\}$ has a $(1, 12N_V)$-efficient representation.

The simulator then outputs $(x, r, \widehat{W}, a)$ where r is the randomness used by cheating referee $\widehat{R}^{na}$. By construction, this output is distributed identically to $\mathrm{View}(\widehat{R}^{na}(x) \leftrightarrow \mathcal{S}_{ZK}(x))$. □

4.2 General cheating referees

We now show that if that for an arbitrary cheating referee $\widehat{R}$, there exists simulator whose output is distributed according to $\mathrm{View}(\widehat{R}(x) \leftrightarrow \mathcal{S}_{ZK}(x))$.

As mentioned earlier, the difficulty is that $\widehat{R}$ could send questions to a set of players, and then depending on their answers, adaptively choose questions for another set of players, and so on. The arguments from Section 4.1 strongly rely on the fact that the simulator can sample all of the questions before sampling the answers. In this section, we show how to simulate the interaction between the referee and the players in the adaptive scenario.

Lemma 20. *For every PPT $\widehat{R}$, there exists a PPT simulator $\mathrm{Sim}_{\widehat{R}}$ such that the output distribution of $\mathrm{Sim}_{\widehat{R}}(x)$ is equal to $View(\widehat{R}(x) \leftrightarrow \mathcal{S}_{ZK}(x))$.*

Proof. A general cheating referee $\widehat{R}$ behaves as follows: using randomness, it samples a set of players $B_1 \subseteq \mathcal{P} = \{(D, r) : D \in \{V, P\}, r \in [N_D]\}$, followed by some questions $\widehat{W}^{B_1}$ for those players. It sends $\widehat{W}^{B_1}$ to the B_1 players, and receives a partial answer vector a^{B_1}. Based on its randomness and the answers received, the referee samples another set of players $B_2 \subseteq \mathcal{P} \setminus B_1$ and questions

$\widehat{W}^{B_2}$ for the B_2 players. We assume that B_2 is disjoint from B_1 because the players would abort the protocol if they are interacted more than once. The referee continues in this manner until it halts.

The general simulator Sim runs the referee $\widehat{R}$ on randomness s to obtain the sample $(B_1, \widehat{W}^{B_1})$. To simulate the B_1 players' responses to $\widehat{W}^{B_1}$, the simulator will arbitrarily complete $\widehat{W}^{B_1}$ to a question tuple $\widehat{W}_1$ for all players, and then call $\mathrm{SimDensity}$ on $\widehat{W}_1$ to obtain a density matrix ρ_1 defined on registers $S_{\widehat{W}_1}$. With this density matrix, the simulator Sim can sample a partial answer vector a^{B_1} with probability $\mathrm{Tr}(\mathsf{Enc}_{outer}(\rho_1)\,\widehat{W}^{B_1}(a^{B_1}))$. This partial answer vector can be sampled in the same way as described in the simulation for the non-adaptive referee in Lemma 19. Note that this distribution does not depend how the question tuple $\widehat{W}^{B_1}$ was completed, since the distribution is non-signalling.

Based on this sampled answer vector a^{B_1} and the randomness s, the simulator can continue executing $\widehat{R}$ to obtain a sample $(B_2, \widehat{W}^{B_2})$. The simulator then constructs a question tuple $\widehat{W}_2$ that contains both $\widehat{W}^{B_1}$ and $\widehat{W}^{B_2}$ (which are question tuples to disjoint sets of players), and invokes $\mathrm{SimDensity}$ to efficiently compute a density matrix ρ_2 defined on registers $S_{\widehat{W}_2}$. The simulator can then sample a partial answer vector a^{B_2} with probability

$$\frac{\mathrm{Tr}\left(\widehat{W}^{B_2}(a^{B_2}) \otimes \widehat{W}^{B_1}(a^{B_1})\,\mathsf{Enc}_{outer}(\rho_2)\right)}{\mathrm{Tr}(\mathsf{Enc}_{outer}(\rho_2)\,\widehat{W}^{B_1}(a^{B_1}))}.$$

Once again, this partial answer vector can be sampled in the same way as described in the proof of Lemma 19. In the end, the simulator can repeat this process and obtain a sequence $(x, s, \widehat{W}^{B_1}, a^{B_1}, \widehat{W}^{B_2}, a^{B_2}, \ldots)$ that is distributed identically to $\mathrm{View}(\widehat{R}(x) \leftrightarrow \mathcal{S}_{ZK}(x))$.

The complete simulation algorithm is described in detail in Figure 3. It is easy to see that the simulator runs in polynomial time. $\square$

```
Algorithm: $\mathrm{Sim}_{\widehat{R}}(\mathrm{x})$

1. Set $i = 1$.
2. Sample randomness $s$ for $\widehat{R}$.
3. Set $\pi = (x, s)$.
4. While $\widehat{R}$ has not halted:
   (a) Continue the execution of the referee $\widehat{R}$ on randomness $s$, the previous $i-1$ samples
       $(B_1, \widehat{W}^{B_1}, a^{(B_1)}), \ldots, (B_{i-1}, \widehat{W}^{B_{i-1}}, a^{(B_{i-1})})$, to obtain a new sample $(B_i, \widehat{W}^{B_i})$. If $B_i$ has
       non-zero intersection with any of the $B_1, \ldots, B_{i-1}$, add abort to the end of $\pi$ and output
       $\pi$.
   (b) Let $\widehat{W}_i$ denote the question tuple that is the concatenation of $\widehat{W}^{B_1}, \widehat{W}^{B_2}, \ldots, \widehat{W}^{B_i}$ with
       arbitrary questions to the players in $\mathcal{P} \setminus (B_1 \cup \cdots \cup B_i)$.
   (c) Execute SimDensity on input $\widehat{W}_i$ to obtain a $(O(T), O(1))$-efficient representation of the
       density matrix $\rho_i$ supported on registers $S_{\widehat{W}_i}$.
   (d) Sample $a^{B_i}$ with probability
       $$\frac{\mathrm{Tr}\left(\Pi_{i-1} \otimes \widehat{W}^{B_i}(a^{B_i})\, \mathsf{Enc}_{outer}(\rho_i)\right)}{\mathrm{Tr}\left(\Pi_{i-1}\, \mathsf{Enc}_{outer}(\rho_i)\right)}$$
       where
       $$\Pi_{i-1} = \widehat{W}^{B_1}(a^{B_1}) \otimes \cdots \otimes \widehat{W}^{B_{i-1}}(a^{B_{i-1}}).$$
   (e) Add $(\widehat{W}^{B_i}, a^{B_i})$ to the end of $\pi$.
   (f) Set $i = i + 1$.
5. Output $\pi$.
```

Figure 3: The simulator $\mathrm{Sim}_{\widehat{R}}$

4.3 Simulating snapshots

We now prove Lemma 17. For convenience we recall the Lemma statement.

Lemma 17. *There exists a PPT algorithm* SimSnapshot *that on input* (x, Y, t) *such that*

1. *x is a binary string*
2. *Y is a subset of registers used in the honest strategy $\mathcal{S}_{ZK}(x)$ (that does not include the prover registers* P *nor the prover flags* F*) that has size at most $4L$, and*
3. *t is an integer between 0 and the length of the protocol circuit $V_{enc}(x)$*

outputs matrix entries of the density matrix

$$\mathrm{Tr}_{\overline{Y}}\left(|\Delta_t\rangle\langle\Delta_t|\right)$$

where $|\Delta_t\rangle$ *is defined as in* (5).

Proof. Fix the protocol circuit $V_{enc} = V_{enc}(x)$. For convenience, we omit mention of the input x for the remainder of this proof. Let T denote the length of the circuit V_{enc}. We notice that our parameters imply that $4L = 192$, and therefore $\mathcal{C}_{inner}$ is a $4L$-simulatable code and we denote m as the blocklength of this code, as defined in Theorem 6.

The protocol circuit acts on registers $\mathsf{A}, \mathsf{B}, \mathsf{O}, \mathsf{N}, \mathsf{M}, \mathsf{P}$. Since the set Y does not include any subregister of the prover register P, we only consider the subregisters of $\mathsf{R} = \mathsf{ABONM}$. At each time t, we say that a group of m qubit registers $\mathsf{R}_{i_1}, \ldots, \mathsf{R}_{i_m}$ form an *encoded block* if and only if $\Pi|\Delta_t\rangle = |\Delta_t\rangle$ where Π is the projector onto the codespace for the qubits $\mathsf{R}_{i_1}, \ldots, \mathsf{R}_{i_m}$. Since the protocol circuit $V_{enc}(x)$ can be computed in polynomial time, determining the encoded block of qubits than a physical qubit belongs to can be efficiently done.

As explained in Section 3.1.1, we split the phases of the circuit into micro-phases. For every time $t \in \{0, 1, 2, \ldots, T\}$, let $start(t) \leq t$ denote the start of the micro-phase containing time t, and let $end(t) \geq t$ denote the end of the micro-phase containing time t. For each time t, we can partition the qubit subregisters into three categories:

- **Active**: These are qubits that have been acted upon by a gate $g_{t'}$ for some time $t' \in \{start(t), \ldots, t\}$. Let $\mathcal{A}(t)$ denote the set of active qubit registers at time t.
- **Encoded qubits**: These are qubits that belong to an encoded block, and are not active. Let $\mathcal{E}(t)$ denote the set of encoded qubit registers at time t.
- **Unencoded qubits**: These are unencoded ancilla qubits in the state $|0\rangle$ or in the state $|1\rangle$, and are not active. Let $\mathcal{U}_0(t)$ and $\mathcal{U}_1(t)$ denote the sets of unencoded qubit registers in the state $|0\rangle$ an $|1\rangle$, respectively, at time t.

Unencoded qubits are in a "known" state throughout the entire circuit V_{enc} in the sense that their state is independent of the input x. In fact, for all t the state $|\Delta_t\rangle$ can be written as

$$|\Delta_t\rangle = |\Sigma_t\rangle_{\mathcal{A}(t)\mathcal{E}(t)} \otimes |0\cdots 0\rangle_{\mathcal{U}_0(t)} \otimes |1\cdots 1\rangle_{\mathcal{U}_1(t)}$$

where $|\Sigma_t\rangle$ corresponds to the registers that are either active or encoded, and the remaining qubits are unencoded ancillas.

By construction, the protocol circuit V_{enc} satisfies the following invariant: at the beginning and end of every micro-phase of the circuit, all qubit subregisters are either encoded, or unencoded. Qubits can only be active within a micro-phase.

We now argue that the description of $\mathrm{Tr}_{\overline{Y}}(|\Delta_t\rangle\langle\Delta_t|)$ can be efficiently computed for all t. We argue this for each micro-phase separately. Let $t_0 = start(t)$.

Idling phase During an idling phase, all qubits are either encoded or unencoded, and none are active. The reduced density matrix $\mathrm{Tr}_{\overline{Y}}(|\Delta_t\rangle\langle\Delta_t|)$ thus consists of either at most $|Y|$ unencoded $|0\rangle$ and $|1\rangle$ ancilla qubits, and the reduced density matrix of some encoded blocks on at most $|Y| \leq 4L$ qubits. By Theorem 6, the reduced density matrix of the encoded blocks is efficiently computable, and thus $\mathrm{Tr}_{\overline{Y}}(|\Delta_t\rangle\langle\Delta_t|)$ is efficiently computable.

Resource encoding In a resource encoding phase, a constant number of unencoded ancilla bits in $|\Delta_{t_0}\rangle$ will be transformed into an encoded resource state in $|\Delta_{end(t)}\rangle$, and the rest of the qubits are either in an encoded block or unencoded ancilla qubits. Thus the reduced density matrix $\mathrm{Tr}_{\overline{Y}}(|\Delta_t\rangle\langle\Delta_t|)$ is a tensor product of the reduced density matrix of some encoded blocks (which is efficiently computable by Theorem 6), unencoded ancilla qubits, and the reduced density matrix of the intermediate state of a resource encoding circuit acting on a constant number of ancillas (which is efficiently computable). Thus $\mathrm{Tr}_{\overline{Y}}\left(|\Delta_t\rangle\langle\Delta_t|\right)$ is efficiently computable.

Logical operation In a logical operation micro-phase, either a logical Hadamard, logical CNOT, or logical Toffoli are being implemented on some encoded code blocks as well as some unencoded ancilla qubits. Let $U \in \{H, \Lambda(X), \Lambda^2(X)\}$ be the logical gate, and $O_1, O_2, \ldots, O_{t-t_0}$ denote the first $t - t_0$ gates of the encoding of U. We have that

$$|\Delta_t\rangle = O_{t-t_0} \cdots O_1 |\Delta_{t_0}\rangle.$$

Since all qubits of $|\Delta_{t_0}\rangle$ are correctly encoded, this corresponds to the simulation in the middle of the application of a logical gate, and again by Theorem 6, $\mathrm{Tr}_{\overline{Y}}(|\Delta_t\rangle\langle\Delta_t|)$ can also be efficiently computable.

Output decoding In the honest strategy $\mathcal{S}_{ZK}(x)$, the state $|\Delta_{t_0}\rangle$ can be written as a tensor product

$$|\Delta_{t_0}\rangle = \mathsf{Enc}_{inner}(|1\rangle)_{\mathsf{O}} \otimes |\Sigma_{t_0}\rangle_{\mathcal{E}(t_0)} \otimes |0\cdots 0, 1\cdots 1\rangle_{\mathcal{U}(t_0)}.$$

This is because by assumption the strategy $\mathcal{S}_{ZK}(x)$ causes the protocol circuit V_{enc} to accept with probability 1, and therefore the register O at the beginning of the Output Decoding phase will store an encoding of $|1\rangle$.

Therefore, the reduced density matrix $\mathrm{Tr}_{\overline{Y}}\left(|\Delta_t\rangle\langle\Delta_t|\right)$ is a tensor product of the reduced density matrix of a decoding circuit acting on $\mathsf{Enc}_{inner}(|1\rangle)$ (which is efficiently computable), the reduced density matrix of $|\Sigma_{t_0}\rangle$ on at most $|Y| \leq 4L$ qubits (which is efficiently computable by Theorem 6), and a constant number of unencoded ancilla qubits. Thus $\mathrm{Tr}_{\overline{Y}}\left(|\Delta_t\rangle\langle\Delta_t|\right)$ is efficiently computable. □

4.4 Completing the proof of Theorem 1

If the completeness and soundness of the original MIP* protocol for L are 1 and s respectively, then the soundness of the resulting zero knowledge protocol V_{ZK} for L has completeness 1 (i.e. perfect completeness) and has soundness s' that is polynomially related to $1 - s$:

$$s' \leq 1 - \frac{(1-s)^\beta}{p(n)}$$

for some universal constant β and polynomial p.

Our zero knowledge transformation is not immediately gap preserving, in the sense that if $1 - s$ is a constant, the new soundness s' is only separated from 1 by an inverse polynomial. Since the standard definition of the complexity classes QMIP, MIP*, and PZK-MIP* have constant completeness-soundness gaps, our result does not immediately show that MIP* $\subseteq$ QMIP $\subseteq$ PZK-MIP*.

To remedy this, we employ the gap amplification techniques described in Section 2.5. Suppose that the soundness s' of V_{ZK} is at most $1 - 1/q$ for some polynomial q. First, we apply the anchoring transformation to V_{ZK} to obtain a new protocol $V_{ZK,\perp}$ such that

$$\omega^*(V_{ZK,\perp}) = \alpha + (1-\alpha)\omega^*(V_{ZK})$$

for some constant α. Then, we use Theorem 11 of Bavarian, Vidick and Yuen [3] to argue that the parallel repetition of $V_{ZK,\perp}$ has the desired soundness properties. In the case that $\omega^*(V_{ZK}) = 1$, then $\omega^*(V^m_{ZK,\perp}) = 1$ for all m. Otherwise, for some polynomial m that depends on q, k, α, and V_{ZK}, we have that $\omega^*(V^m_{ZK,\perp}) \leq 1 - (1-s)^\gamma$ for some universal constant γ. Thus, the soundness of $V^m_{ZK,\perp}$ is polynomially related to the original completeness-soundness gap $1 - s$, and it also decides L.

It remains to argue that the amplified protocol $V^m_{ZK,\perp}$ still has the perfect zero knowledge property. In general, this is a delicate issue, since it is known that parallel repetition does not preserve zero knowledge in a black box manner [20, 5, 33].

In our case, however, since the referee is constrained to interacting with each prover only once, we can simulate the interaction in the amplified protocol $V^m_{ZK,\perp}$ by essentially running many copies of the simulator $\mathrm{Sim}_{\widehat{R}}$ described in Figure 3 in parallel. Notice that the honest strategy for $V^m_{ZK,\perp}$ consists of sharing m copies of the history state, and performing independent measurements on each of these copies. It is not hard to see that the interaction in $V^m_{ZK,\perp}$ can be simulated efficiently.

The number of provers involved in the protocol executed by $V^m_{ZK,\perp}$ is $k+4$, and the protocol is 1-round, which implies that

$$L \in \mathsf{PZK\text{-}MIP}^*_{1,s''}[k+4, 1].$$

where $s'' \leq 1 - (1-s)^\gamma$. This concludes the proof of Theorem 1.

5 Simulatable codes

In this section we show the existence of simulatable codes. We start by introducing stabilizer codes and some notation in Section 5.1. Then, we analyse low-weight measurements on codewords of a stabilizer QECC in Section 5.2. In Sections 5.3 and 5.4 we show how to simulate low-weight measurements on the encoding of transversal and non-transversal gates, respectively. Finally, in Section 5.5 we show that the concatenated Steane code is a simulatable code.

5.1 Stabilizer codes

We present some preliminary background on stabilizer codes, an important class of QECCs. For an in-depth reference on stabilizer codes, we recommend consulting [22].

Let $\mathcal{P}_n$ be the n-qubit Pauli group, so P_n is the set of n-qubit unitaries $W_1 \otimes \cdots \otimes W_n$, where $W_i \in \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}$ for all $i = 1, \ldots, n$. The *weight* of an element $W_1 \otimes \cdots \otimes W_n \in P_n$ is $|\{1 \leq i \leq n : W_i \notin \{\pm I, \pm iI\}|$.

An $[[n, k]]$ stabilizer code is specified by an abelian subgroup $\mathcal{S} \subseteq P_n$ such that $-I \notin \mathcal{S}$, and any minimal generating set of $\mathcal{S}$ has size $n - k$. Usually we fix a minimal generating set $g_1, \ldots, g_{n-k}$ of $\mathcal{S}$, and refer to these elements as the stabilizers of the code. The codespace of an $[[n, k]]$ stabilizer code $\mathcal{S}$ is the subspace of vectors in $\mathbb{C}^{2^n}$ fixed by $\mathcal{S}$. In other words, $|\psi\rangle$ is in the code if and only if $g|\psi\rangle = |\psi\rangle$ for all $g \in \mathcal{S}$. This space always has dimension 2^k, and hence an $[[n, k]]$ stabilizer code can encodes k-qubit states in n-qubit states. To fix one of all the possible encodings of k-qubit states, we can find $\overline{Z}_1, \ldots, \overline{Z}_k \in \mathcal{P}_n$ such that $\mathcal{S}' = \langle g_1, \ldots, g_{n-k}, \overline{Z}_1, \ldots, \overline{Z}_k\rangle$ is abelian and is minimally generated by $g_1, \ldots, g_k, \overline{Z}_1, \ldots, \overline{Z}_k$. The encoding then sends the computational basis state $|x_1 \cdots x_k\rangle$ to the unique state in the codespace fixed by $g_1, \ldots, g_k, (-1)^{x_1}\overline{Z}_1, \ldots, (-1)^{x_k}\overline{Z}_k$.

More generally, an $[[n, k]]$ stabilizer code can be used to encode mk-qubit states in mn-qubit states. This can be expressed in the stabilizer formalism by taking the product of the stabilizer code with itself m times. Specifically Let $\Delta^i_{n,mn} : \mathcal{P}_n \to \mathcal{P}_{mn}$ be the inclusion[10] induced by having $\mathcal{P}_n$ act on qubits $(i-1)n + 1, (i-1)n + 2, \ldots, in$ of an mn-qubit register. For example, we have

[10]An inclusion map $f : A \mapsto B$ consists in treating an element $x \in A$, as an element of B.

$\Delta^2_{2,6}(X \otimes Z) = I \otimes I \otimes X \otimes Z \otimes I \otimes I$. When n and m are clear, we write Δ^i for $\Delta^i_{n,mn}$. Given a stabilizer code $\mathcal{S}$ with stabilizers $g_1, \ldots, g_{n-k}$ in $\mathcal{P}_n$, let

$$\mathcal{S}^{\otimes m} := \langle \Delta^i(g_j) \text{ where } 1 \leq i \leq m, 1 \leq j \leq n-k \rangle.$$

This defines an $[[mn, mk]]$ stabilizer code with minimal generating set $\{\Delta^i(g_j)\}$. To encode mk-qubit states in this code, we can take elements $\overline{Z}_1, \ldots, \overline{Z}_k \in \mathcal{P}_n$ specifying an encoding of $\mathcal{S}$ as above. Then elements $\Delta^i(\overline{Z}_j)$, $1 \leq i \leq m$, $1 \leq j \leq k$ specify an encoding for the code $\mathcal{S}^{\otimes m}$.

Every pair of elements $g, h \in \mathcal{P}_n$ either commute or anticommute. Since stabilizer codes do not contain $-I$ by definition, the normalizer $N(\mathcal{S})$ of a stabilizer code $\mathcal{S}$ in $\mathcal{P}_n$ is the set of all elements of $\mathcal{P}_n$ which commute with every element in $\mathcal{S}$. The *distance* of a stabilizer code is the smallest integer d such that $N(\mathcal{S}) \setminus \mathcal{S}$ contains an element of weight d. An $[[n, k, d]]$ stabilizer code is an $[[n, k]]$ stabilizer code of distance $\geq d$.

5.2 Computing partial trace of codewords

Suppose we want to the compute the partial trace $\mathrm{Tr}_{\overline{Q}}(\rho)$ for some n-qubit state ρ and subset of qubits Q. Because $\mathcal{P}_{|Q|}$ contains an orthogonal basis (in the Hilbert-Schmidt inner product) for $2^{|Q|} \times 2^{|Q|}$ matrices and is closed under the adjoint operation, it is sufficient to compute the inner products $\mathrm{Tr}(\mathrm{Tr}_{\overline{Q}}(\rho) w)$ for all elements $w \in \mathcal{P}_{|Q|}$. Extending the notation from Section 5.1, let $\Delta^Q_n : \mathcal{P}_{|Q|} \to \mathcal{P}_n$ be the inclusion induced by having elements of $\mathcal{P}_{|Q|}$ act on qubits Q (so for instance, $\Delta^i_{n,mn} = \Delta^Q_{mn}$ where $Q = \{(i-1)n + 1, \ldots, in\} \subseteq \{1, \ldots, mn\}$). Then

$$\mathrm{Tr}\left(\mathrm{Tr}_{\overline{Q}}(\rho) w\right) = \mathrm{Tr}(\rho \Delta^Q_n(w)),$$

so to compute $\mathrm{Tr}_{\overline{Q}}(\rho)$, it is sufficient to be able to compute $\mathrm{Tr}(\rho \Delta^Q_n(w))$ for all $w \in \mathcal{P}_{|Q|}$. We record this fact in the following lemma:

Lemma 21. *The partial traces $\mathrm{Tr}_{\overline{Q}}(\rho)$ of an n-qubit state ρ can be computed from the traces $\mathrm{Tr}(\rho \Delta^Q_n(w))$, $w \in \mathcal{P}_{|Q|}$, in time $\exp(O(|Q|))$.*

For a stabilizer code $\mathcal{S}$, we can easily compute $\mathrm{Tr}(\mathsf{Enc}(\rho) w)$, without knowing ρ, as long as w is not in $N(\mathcal{S}) \setminus \mathcal{S}$.

Lemma 22. *Let $\mathsf{Enc}(\rho)$ be an encoding of a k-qubit state ρ in an $[[n, k]]$ stabilizer code $\mathcal{S}$, and suppose $w \notin N(\mathcal{S}) \setminus \mathcal{S}$. Then*

$$\mathrm{Tr}(\mathsf{Enc}(\rho) w) = \begin{cases} 1 & w \in \mathcal{S} \\ 0 & w \notin \mathcal{S} \end{cases}.$$

Proof. If $w \in \mathcal{S}$, then $\mathsf{Enc}(\rho) w = \mathsf{Enc}(\rho)$ by definition, since w fixes the codespace of $\mathcal{S}$. So $\mathrm{Tr}(\mathsf{Enc}(\rho) w) = \mathrm{Tr}(\mathsf{Enc}(\rho)) = 1$.

Suppose $w \notin N(\mathcal{S})$. Let $g_1, \ldots, g_{n-k}$ be a minimal generating set for $\mathcal{S}$. By a standard argument, we can assume that $g_2, \ldots, g_{n-k}$ commute with w, and g_1 anticommutes. Let

$$P = \left(\frac{I + g_1}{2}\right)\left(\frac{I + g_2}{2}\right) \cdots \left(\frac{I + g_{n-k}}{2}\right),$$

the projection onto the codespace of $\mathsf{Enc}(\rho)$. Then $Pw = wP'$, where

$$P' = \left(\frac{I - g_1}{2}\right)\left(\frac{I + g_2}{2}\right) \cdots \left(\frac{I + g_{n-k}}{2}\right),$$

an orthogonal projection to P. So

$$\mathrm{Tr}(\mathsf{Enc}(\rho)w) = \mathrm{Tr}(P\mathsf{Enc}(\rho)Pw) = \mathrm{Tr}(P\mathsf{Enc}(\rho)wP') = \mathrm{Tr}(P'P\mathsf{Enc}(\rho)w) = 0. \qquad \square$$

In particular, if $\mathcal{S}$ is an $[[n,k,d]]$ stabilizer code, and $|Q| < d$, then $\Delta_n^Q(w)$ will have weight $< d$ for all $w \in \mathcal{P}_{|Q|}$, and hence $\mathrm{Tr}(\mathsf{Enc}(\rho)\Delta_n^Q(w))$ will be equal to 1 or 0 for all $w \in \mathcal{P}_Q$, depending on whether $\Delta_n^Q(w) \in \mathcal{S}$. If $\mathcal{S}$ is non-degenerate, meaning that every element of $\mathcal{S}$ has weight $\geq d$, then $\mathrm{Tr}(\mathsf{Enc}(\rho)\Delta_n^Q(w)) = 0$ unless $\Delta_n^Q(w)$ is the identity matrix, so $\mathrm{Tr}_{\overline{Q}}(\mathsf{Enc}(\rho))$ will be maximally mixed. If the code is degenerate, $\mathrm{Tr}_{\overline{Q}}(\mathsf{Enc}(\rho))$ will not always be maximally mixed; instead, it is maximally mixed over the invariant subspace of the degenerate stabilizers.

If $\mathcal{S}$ is an $[[n,k,d]]$ stabilizer code, then the product code $\mathcal{S}^{\otimes m}$ only has distance d (and hence is an $[[mn,mk,d]]$ code). However, we can say more about when an element of $\mathcal{P}_{mn}$ is in $N(\mathcal{S}^{\otimes m})$.

Lemma 23. *Let $\mathcal{S}$ be an $[[n,k,d]]$ code, and suppose $w_1,\ldots,w_m$ are elements of $\mathcal{P}_n$. Then $w = w_1 \otimes \cdots \otimes w_m \in \mathcal{P}_{mn}$ belongs to $N(\mathcal{S}^{\otimes m})$ if and only if $w_i \in N(\mathcal{S})$ for all $1 \leq i \leq m$. In particular, if w_i has weight $< d$ for every $1 \leq i \leq m$, then $w \in N(\mathcal{S}^{\otimes m})$ if and only if $w \in \mathcal{S}^{\otimes m}$.*

Proof. Suppose $w \in N(\mathcal{S}^{\otimes m})$. Then $w\Delta^i(g)w^{-1} = \Delta^i(g)$ for all $g \in \mathcal{S}$. But $w = \Delta^1(w_1)\cdots\Delta^m(w_m)$, so $w\Delta^i(g)w^{-1} = \Delta^i(w_i g w_i^{-1})$. Since Δ^i is an inclusion, $w_i \in N(\mathcal{S})$.

If w_i has weight $< d$ for all $1 \leq i \leq m$, and $w \in N(\mathcal{S}^{\otimes m})$, then we must have $w_i \in \mathcal{S}$ for all $1 \leq i \leq m$, and hence $w \in \mathcal{S}^{\otimes m}$. $\square$

5.3 Simulatable encoding of transversal Clifford gates

We now consider what happens if we add operations on encoded states into the picture. For simplicity of description, we restrict to $[[n,1,d]]$ stabilizer codes. Recall that the n-qubit Clifford group $\mathcal{C}_n$ is the normalizer of $\mathcal{P}_n$ in the group of unitaries.

Lemma 24. *Let $\mathcal{S}$ be an $[[n,1,d]]$ stabilizer code, let ρ be a k-qubit state, and let $O_1,\ldots,O_\ell \in \mathcal{C}_{nk}$ such that O_i acts on a subset Q_i of the physical qubits of $\mathcal{S}^{\otimes k}$, where $Q_i \cap Q_j = \emptyset$ for all $1 \leq i \neq j \leq \ell$, and Q_i contains at most one physical qubit from each logical qubit of $\mathcal{S}^{\otimes k}$ for all $1 \leq i \leq \ell$.*

If Q is a subset of the physical qubits of $\mathcal{S}^{\otimes k}$ with $|Q| < d$, then we can compute

$$\mathrm{Tr}\left(O_\ell \cdots O_1 \mathsf{Enc}(\rho)(O_\ell\cdots O_1)^\dagger \Delta_{nk}^Q(w)\right)$$

for all $w \in \mathcal{P}_{|Q|}$ without knowledge of ρ. Furthermore, if n and d are constant, then this computation can be done in polynomial time in k, ℓ, and the maximum amount of time needed to compute $O_i^\dagger g O_i \in \mathcal{P}_{nk}$ for any $1 \leq i \leq \ell$ and $g \in \mathcal{P}_{nk}$.

Proof. Since $O_\ell \cdots O_1 \in \mathcal{C}_{nk}$,

$$(O_\ell\cdots O_1)^\dagger \Delta_{nk}^Q(w) = w'(O_\ell\cdots O_1)^\dagger,$$

where $w' \in \mathcal{P}_{nk}$ can be computed in time polynomial in ℓ and the time needed to compute $O_i^\dagger g O_i$ for any $g \in \mathcal{P}_{nk}$ and $1 \leq i \leq \ell$.

Let $R = Q \setminus \bigcup_{i=1}^\ell Q_i$. Write $w' = W_1 \otimes \cdots \otimes W_{nk}$ where $W_j \in \mathcal{P}_1$ for all $1 \leq j \leq nk$, and let $w_a = W_{(a-1)n+1} \otimes \cdots \otimes W_{an}$, so w_a contains the operators corresponding to the ath logical qubit, $1 \leq a \leq k$. Since the operators O_i act on disjoint sets of physical qubits, if $W_j \notin \{\pm I, \pm iI\}$, then either $j \in R$, or $j \in Q_i$ for some i with $Q_i \cap Q \neq \emptyset$. Because

$$|\{(a-1)n+1,\ldots,an\} \cap Q_i| \leq 1 \text{ for all } 1 \leq i \leq \ell,$$

we see that the weight of w_a is at most

$$|\{1 \leq i \leq \ell : Q_i \cap Q \neq \emptyset\}| + |R| \leq |Q| < d.$$

By Lemma 23, $w' \in \mathcal{S}^{\otimes k}$ if and only if $w' \in N(\mathcal{S}^{\otimes k})$. Also, $w' \in N(\mathcal{S}^{\otimes k})$ if and only if $w_a \in N(\mathcal{S})$ for all $1 \leq a \leq \ell$. Since $\mathcal{S}$ is fixed, we can check whether $w_a \in N(\mathcal{S})$ in constant time, and hence we can determine whether $w' \in \mathcal{S}^{\otimes k}$.

Finally, we have

$$\mathrm{Tr}\left(O_\ell \cdots O_1 \mathsf{Enc}(\rho)(O_\ell \cdots O_1)^\dagger \Delta^Q_{nk}(w)\right) = \mathrm{Tr}\left(O_\ell \cdots O_1 \mathsf{Enc}(\rho) w' (O_\ell \cdots O_1)^\dagger\right) = \mathrm{Tr}(\mathsf{Enc}(\rho) w').$$

Since $w' \notin N(\mathcal{S}^{\otimes k}) \setminus \mathcal{S}^{\otimes k}$,

$$\mathrm{Tr}(\mathsf{Enc}(\rho) w') = \begin{cases} 1 & w' \in \mathcal{S}^{\otimes m} \\ 0 & w' \notin \mathcal{S}^{\otimes m}. \end{cases}$$

by Lemma 22. □

With the previous lemma, we can show how to simulate the transversal encoding of Clifford gates.

Proposition 25. *If the $[[n, 1, d]]$ stabilizer code accepts a transversal encoding of a k-qubit Clifford gate G, then such encoding is s simulatable for all $s < d$.*

Proof. Let ρ be an n-qubit state, $\underline{a} = (a_1, ..., a_k)$ be a k-tuple of disjoint integers between 1 and n and $O_1(\underline{a}, \ldots, O_\ell(\underline{a})$ be the encoding of $G(\underline{a})$ for $\ell = n$, and let S be a subset of $\{1, \ldots, \ell n\}$ with $|S| \leq s$.

If $1 \leq t \leq \ell$, then the set of gates $O_1(\underline{a}), \ldots, O_t(\underline{a})$ satisfy the conditions of Lemma 24, and furthermore, since $O_i(\underline{a})$ acts on at most k physical qubits, $O_i(\underline{a})^\dagger w O_i(\underline{a})$ can be computed in polynomial time in n for all $w \in \mathcal{P}_{\ell n}$.

The proposition follows from Lemmas 21 and 24. □

5.4 Simulatable encoding of non-transversal gates

It is well known that there is no QECC where all logical operations from a universal set of gates can be performed transversally. In order to circumvent this barrier, we can use other tools from fault-tolerant quantum computation, namely magic states.

The general procedure for applying a k-qubit gate G using a magic state for it is depicted in Figure 4. The input to this procedure is some k-qubit state ρ, on which we want to apply G, and the magic state $|\mathsf{Magic}_G\rangle$. In the first phase, a unitary V_0 is applied to both registers. In the second phase, some of the qubits are measured. Finally, classical controled unitaries V_i are applied. We assume for simplicity that each V_i can be applied transversally [11]. The output of this procedure is then $G\rho G^\dagger$.

[11]More generally, we could assume that V_i can be decomposed on gates that can be applied transversally and then the encoding of V_i consists of the sequence of encoding for each of these gates.

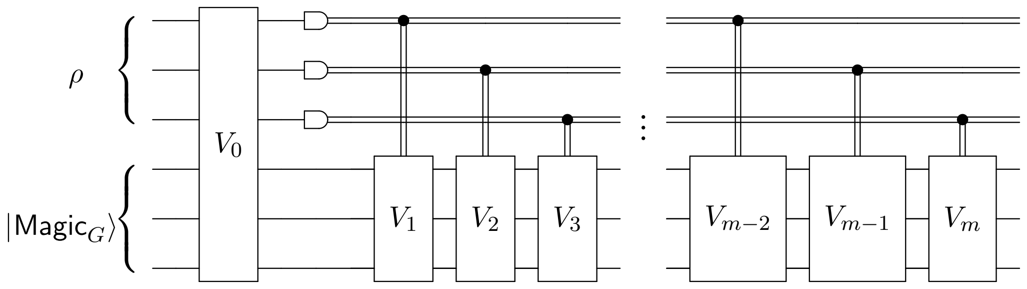


Figure 4: Teleportation gadget

There are two issues when one try to prove that the encoding of such gadget is simulatable. First, it would be necessary to *decode* the measured qubit to perform the controlled unitary, and this could make the encoding non-simulatable. In order to solve this problem, we present in Section 5.4.1 an extra property that we require from our QECC and how it allows us to define the non-transversal encoding of G. Secondly, measurements are not allowed in Definition 5. In this case, we show how to perform a coherent encoding of gate G in Section 5.4.2.

With these two pieces in hand, we are able to show the simulation of the coherent non-transversal encoding of G in Section 5.4.3.

5.4.1 Simulatable encoding of classically-controlled transversal gates

As mentioned before, one of the possible issues when simulating the encoding of Figure 4 is that the control qubit must be decoded during the gadget. Here, we show how to workaround this issue by assuming that the QECC has the following property.

Definition 26. *For $b \in \{0,1\}$, let E_b be the set of strings in $\mathsf{Enc}(|b\rangle)$. A QECC $\mathcal{S}$ is order-consistent with a set of unitaries $\mathcal{V}$ if for every $V \in \mathcal{V}$ and every $x \in E_b$, it follows that $V^{|x|} = V^b$.*

We require then our QECC to be order-consistent with the set of unitaries $\mathcal{V} = \{V_0, ..., V_m\}$ in Figure 4. In order to explain how the non-transversal encoding works, let us first assume that starting from $\mathsf{Enc}(|b\rangle) \otimes |\psi\rangle$, we want to apply V^b on $|\psi\rangle$ without decoding $|b\rangle$. This operation can be implemented by applying the gates

$$O_1, O_2, \ldots, O_n$$

where O_i is a $\Lambda(V)$-gate with the ith qubit of $\mathsf{Enc}(|b\rangle)$ as the control qubit and $|\psi\rangle$ as target. We depict such encoding in Figure 5.

Notice that this sequence of gates performs the correct logical operation, since after applying $O_1, O_2, \ldots, O_n$, the resulting state is

$$\sum_{x \in E_b} \alpha_x |x\rangle V^{|x|} |\psi\rangle, \tag{15}$$

where E_b is the set of string in the support of $\mathsf{Enc}(|b\rangle)$. Since we assume that the QECC is order-consistent with $\mathcal{V}$, we have that Equation (15) is equal to

$$\mathsf{Enc}(|b\rangle) V^b |\psi\rangle.$$

We can extend this idea and have an encoding of $|\psi\rangle$, and then we can just decompose O_i

$$O_i = O_{i,1}, O_{i,2}, \ldots, O_{i,n}$$

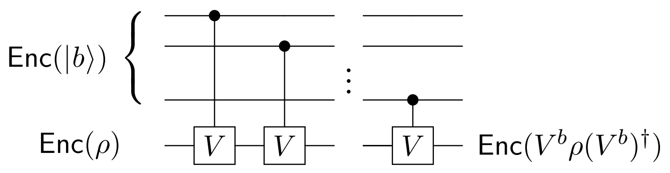


Figure 5: Non-transversal encoding the controlled gates

where $O_{i,j}$ is a $\Lambda(V)$ gate with the ith physical qubit of the encoding of the logical control bit, and the target are the jth physical qubits of the encoding of the logical target state.

With this tool, we are able to prove the encoding of $\Lambda(V)$ is simulatable if the control is a classical known bit.

Lemma 27. *Let $\mathcal{S}$ be a $[[n, 1, d]]$ stabilizer code that is order-consistent with some k-qubit unitary V which has a transversal encoding in $\mathcal{S}$. Then the non-transversal encoding of*

$$\Lambda(V)\left(|x\rangle\langle x| \otimes \rho\right)\Lambda(V)^\dagger$$

for some fixed $x \in \{0,1\}$ is s-simulatable for any $s < d-k$.

Proof. The non-transversal encoding of $\Lambda(V)$ consists of $O_{1,1}, O_{2,1}, \ldots, O_{n,1}, O_{1,2}, \ldots, O_{n,n}$, where $O_{i,j}$ is the $\Lambda(V)$ gate with control on the ith physical qubit of the encoding of the first qubit and jth physical qubits of the logical target qubits.

Let S be the subset of qubits for which we want to compute

$$\mathrm{Tr}_{\overline{S}}(O_{(t_1,t_2)}O_{(t_1-1,t_2)}\cdots O_{2,1}O_{1,1}\mathsf{Enc}(|x\rangle\langle x| \otimes \rho)(O_{(t_1,t_2)}O_{(t_1-1,t_2)}\cdots O_{2,1}O_{1,1})^\dagger),$$

with $|S| \le s$,

$$\mathsf{Enc}(|x\rangle) = \sum_{y\in\{0,1\}^{(k+1)n}} \alpha_y|y\rangle,$$

and

$$\mathsf{Enc}(|x\rangle\langle x| \otimes \rho) = \sum_{y,z\in\{0,1\}^n} \alpha_y\alpha_z^*|y\rangle\langle z| \otimes \mathsf{Enc}(\rho).$$

We can fix the computational basis states $|y\rangle$ and $|z\rangle$ in the support of $\mathsf{Enc}(|x\rangle)$ and the result will follow by linearity. Let $V(j)$ denote the gate V applied to the jth physical qubits of the encodings of each qubit of ρ. We have that

$$\begin{aligned}&O_{(t_1,t_2)}O_{(t_1-1,t_2)}\cdots O_{2,1}O_{1,1}|y\rangle\langle z| \otimes \mathsf{Enc}(\rho)(O_{(t_1,t_2)}O_{(t_1-1,t_2)}\cdots O_{2,1}O_{1,1})^\dagger\\ &= |y\rangle\langle z| \otimes V(t_2)^b V(t_2-1)^x V(t_2-2)^x \cdots V(1)^x \mathsf{Enc}(\rho)\left(V(t_2)^c V(t_2-1)^x V(t_2-2)^x \cdots V(1)^x\right)^\dagger,\end{aligned}$$

where we use the fact that $V(j)^{|y|} = V(j)^x$ for every y in $\mathsf{Enc}(|x\rangle)$ and we define $b = y_1 + \ldots + y_{t_1}$ and $c = z_1 + \ldots + z_{t_1}$. Let $S_1 = S \cap \{1, \ldots, n\}$, and let $S_2 = S \setminus S_1$ (relabelled to be a subset of $\{1, \ldots, kn\}$). Then

$$\begin{aligned}&\mathrm{Tr}_{\overline{S}}(O_{(t_1,t_2)}O_{(t_1-1,t_2)}\cdots O_{2,1}O_{1,1}|y\rangle\langle z| \otimes \mathsf{Enc}(\rho)(O_{(t_1,t_2)}O_{(t_1-1,t_2)}\cdots O_{2,1}O_{1,1})^\dagger)\\ &= \mathrm{Tr}_{\overline{S_1}}(|y\rangle\langle z|) \otimes \mathrm{Tr}_{\overline{S_2}}\left(V(t_2)^b V(t_2-1)^x \cdots V(1)^x \mathsf{Enc}(\rho)\left(V(t_2)^c V(t_2-1)^x V(t_2-2)^x \cdots V(1)^x\right)^\dagger\right).\end{aligned}$$

Because x, y, z and the code are fixed, we can compute $\mathrm{Tr}_{\overline{S_1}}(|y\rangle\langle z|)$ explicitly, so the only remaining step is to compute

$$\mathrm{Tr}_{\overline{S_2}}\left(V(t_2)^b V(t_2-1)^x \cdots V(1)^x \mathsf{Enc}(\rho)\left(V(t_2)^c V(t_2-1)^x V(t_2-2)^x \cdots V(1)^x\right)^\dagger\right).$$

By Lemma 21, it suffices to compute

$$\mathrm{Tr}_{\overline{S_2}}\left(V(t_2)^b V(t_2-1)^x \cdots V(1)^x \mathsf{Enc}(\rho)\left(V(t_2)^c V(t_2-1)^x V(t_2-2)^x \cdots V(1)^x\right)^\dagger \Delta_{kn}^{S_2}(w)\right). \quad (16)$$

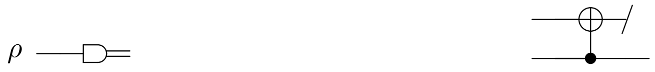
for all $w \in \mathcal{P}_{|S_2|}$. Since x is fixed, and can compute b and c from y and z. Hence if $b = c$, we can compute (16) by Lemma 24, without knowledge of ρ, since $|S_2| \le |S| \le d$. If $b \neq c$, then we can rewrite $V(t_2)^b (V(t_2)^c)^\dagger$ as a linear combination of weight k elements of $\mathcal{P}_{kn}$, meaning that we can compute (16) as long as we can compute

$$\mathrm{Tr}\left(V(t_2-1)^x \cdots V(1)^x \mathsf{Enc}(\rho) V(1)^x \cdots V(t_2-1)^x w'\right)$$

where w' has weight at most $|S_2| + k \le s + k < d$. So once again, we can compute this trace using Lemma 24. □

5.4.2 Coherent encoding

We now present how to remove the measurements from Figure 4. First, we notice that measuring one qubit is equivalent of copying ($\Lambda(X)$) it into a fresh $|0\rangle$ ancilla, and then tracing out this ancilla.



(a) Measurement (b) Coherent measurement

Figure 6: Replacing measurement by coherent measurement

Even though this is sufficient to remove the measurements of Figure 4, this will not be sufficient for the simulation, since we cannot guarantee that the ancilla will be traced out, and it could be hard to keep track of the entangled values. However, if we are just interested in a subset S of qubits with $|S| \le s$ for a fixed s, we can repeat the previous procedure with s ancillas, and in this case, one of such $s+1$ register must be traced out when simulating on S. We depict this in Figure 7.

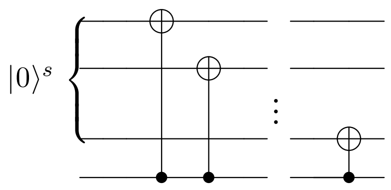


Figure 7: Coherent measurement with more ancilla

5.4.3 Simulation of coherent non-transversal encoding

Starting with Figure 4 and making it non-transversal, as defined in Section 5.4.1, and coherent, as defined in Section 5.4.2, we get the circuit defined in Figure 8.

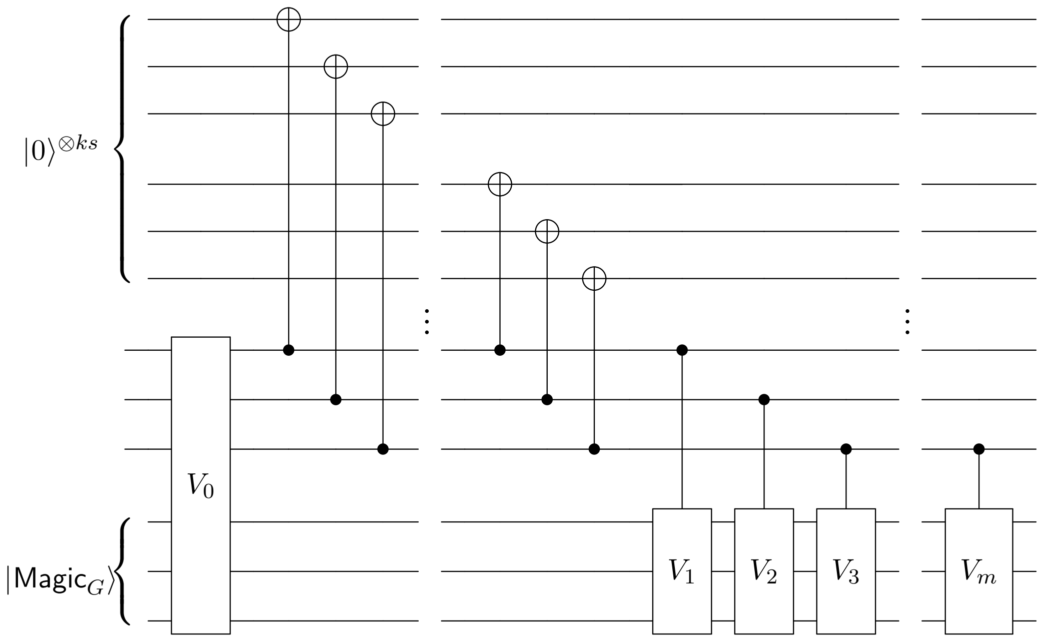


Figure 8: The coherent non-transversal encoding of G

For simplicity, we denote $U_0, \ldots, U_{m+ks}$ the gates of Figure 8 such that $U_0 = V_0$, U_i denotes the $\Lambda(X)$ where the control is the $(i \pmod k)$th input qubit and the target is the ith ancilla qubit for $1 \le i \le ks$; and U_i denotes the controlled-V_{i-ks} for $i > ks$.

By the correctness of the G-gadget of Figure 4, the following lemma holds.

Lemma 28. *Let $U_0, \ldots, U_{m+ks}$ be the quantum gates of the coherent non-transversal encoding of a k-qubit gate G. Let us also assume that for any k-qubit state $|\psi\rangle$, there are states $|\psi_x\rangle$, $x \in \{0,1\}^k$, such that*

$$U_0|\psi\rangle|\mathsf{Magic}_G\rangle = \sum_{x\in\{0,1\}^k} \frac{1}{\sqrt{2^k}}|x\rangle|\psi_x\rangle.$$

It follows that

$$U_{ks}\cdots U_0|\phi\rangle = \sum_{x\in\{0,1\}^k} \frac{1}{\sqrt{2^k}}|x\rangle^{\otimes s+1}|\psi_x\rangle \quad \textit{and} \quad U|\phi\rangle = \sum_{x\in\{0,1\}^k} \frac{1}{\sqrt{2^k}}|x\rangle^{\otimes s+1}G|\psi\rangle.$$

We are now ready to prove that the encoding of the gadget depicted in Figure 8 is simulatable.

Proposition 29. *Let $U_0, \ldots, U_{m+ks}$ be the quantum gates of the coherent non-transversal encoding of a k-qubit gate G. Let also $\mathcal{S}$ be a $[[n,k,d]]$ stabilizer code that is order-consistent with U_i, $0 \le i \le m+ks$. If for any k-qubit state $|\psi\rangle$, there are states $|\psi_x\rangle$, $x \in \{0,1\}^k$, such that*

$$U_0|\psi\rangle|\mathsf{Magic}_G\rangle = \sum_{x\in\{0,1\}^k} \frac{1}{\sqrt{2^k}}|x\rangle|\psi_x\rangle, \tag{17}$$

then the coherent non-transversal encoding of G is s-simulatable for any $s < d-k$.

Proof. Let ρ be a the logical k-qubit state on whose encoding we want to apply the logical gate G and let $\sigma = |0\rangle\langle 0|)^{\otimes ks} \otimes \rho \otimes |\mathsf{Magic}_G\rangle\langle\mathsf{Magic}_G|$ Suppose that each U_i above is encoded by operations

$O_{r_i}, \ldots, O_{r_{i+1}-1}$, where $1 = r_0 < r_1 < \ldots < r_{ks+m} = \ell + 1$. In order to show that the code is simulatable, we need to show how to compute

$$\begin{aligned} &\mathrm{Tr}_S(O_t \cdots O_1 \mathsf{Enc}\,(\sigma)\,(O_t \cdots O_1)^\dagger) \\ &= \mathrm{Tr}_S((O_t \cdots O_{r_i+1})\mathsf{Enc}\left(U_{i-1} \cdots U_0 \sigma (U_{i-1} \cdots U_0)^\dagger\right)(O_t \cdots O_{r_i+1})^\dagger), \end{aligned} \tag{18}$$

for $r_i \leq t \leq r_{i+1}$ and $0 \leq i \leq ks + m$.

By Proposition 25, if $0 \leq i \leq ks$, we can compute the partial trace in Equation (18) for any $|S| < d$ without knowledge of ρ in polynomial time in $|S|$ and $(a_1, ..., a_k)$, since U_i can be applied transversally .

It remains to compute the partial trace in Equation (18) when $ks < i \leq ks + m$. From Equation (17) and Lemma 28, we have

$$\begin{aligned} &\mathrm{Tr}_{\overline{S}}\left(O_t \cdots O_1 \mathsf{Enc}(\sigma)(O_t \cdots O_1)^\dagger\right) \\ &= \sum_{x \in \{0,1\}^k} \frac{1}{2^k} \mathrm{Tr}_{\overline{S}}\left(O_t \cdots O_{r_{ks}} \mathsf{Enc}\left((|x\rangle\langle x|)^{\otimes s+1} \otimes \rho_x\right)(O_t \cdots O_{r_{ks}})^\dagger\right). \end{aligned}$$

for some collection of states ρ_x and we implicitly used the fact that since $|S| \leq s$ and there are $s+1$ registers containing the encoding of x, then there must be a register with an encoding of x that does not overlap with S.

For a fixed x, we can continue to simplify, writing

$$\begin{aligned} &\mathrm{Tr}_{\overline{S}}\left(O_t \cdots O_{r_{ks}} \mathsf{Enc}\left((|x\rangle\langle x|)^{\otimes s+1} \otimes \rho_x\right)(O_t \cdots O_{r_{ks}})^\dagger\right) \\ &= \mathrm{Tr}_{\overline{S}}\left(O_t \cdots O_{r_i} \mathsf{Enc}\left(U_{r_{i-1}} \cdots U_{r_{ks}}(|x\rangle\langle x|)^{\otimes s+1} \otimes \rho_x (U_{r_{i-1}} \cdots U_{r_{ks}})^\dagger\right)(O_t \cdots O_{r_i})^\dagger\right) \\ &= \mathrm{Tr}_{\overline{S}}\left(O_t \cdots O_{r_i} \mathsf{Enc}((|x\rangle\langle x|)^{\otimes s+1} \otimes \rho'_x)(O_t \cdots O_{r_i})^\dagger\right) \end{aligned}$$

for some state ρ'_x, where the equalities follow because $\mathcal{S}$ is order-consistent with all V_i and $U_{r_{ks}}, \ldots, U_{r_{i-1}}$ are only control gates on the first $k(s+1)$ logical qubits.

Thus we only need to compute

$$\mathrm{Tr}_{\overline{S}}\left(O_t \cdots O_{r_i})\mathsf{Enc}((|x\rangle\langle x|)^{\otimes s+1} \otimes \rho'_x)(O_t \cdots O_{r_i})^\dagger\right)$$

for every (known) $x \in \{0,1\}^k$ and (unknown) states ρ'_x, and this can be done efficiently from Lemma 27 if $|S| \leq s < d - k$. In this case,

$$\mathrm{Tr}_{\overline{S}}\left(O_t \cdots O_1 \mathsf{Enc}((|0\rangle\langle 0|)^{\otimes ks} \otimes \rho \otimes |\mathsf{Magic}_G\rangle\langle \mathsf{Magic}_G|)(O_t \cdots O_1)^\dagger\right)$$

is a linear combination of a constant number of matrices which can be computed without knowledge of ρ, in time polynomial in S and 2^k. Hence the encoding is s-simulatable. □

5.5 Explicit construction

In this section, we show that the family of concatenated Steane codes is simulatable. We start by defining the (concatenated) Steane code in Section 5.5.1 and then in Section 5.5.2 we argue that it is simulatable, proving Theorem 6.

5.5.1 Steane code

The Steane code is the $[[7, 1, 3]]$ code stabilized by the following generators:

I	I	I	X	X	X	X
I	X	X	I	I	X	X
X	I	X	I	X	I	X
I	I	I	Z	Z	Z	Z
I	Z	Z	I	I	Z	Z
Z	I	Z	I	Z	I	Z

Table 1: Generators of the Steane code

In particular, the encoding of the basis states are then

$$\mathsf{Enc}(|0\rangle) = \frac{1}{2\sqrt{2}}(|0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle + |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle), \quad (19)$$

and

$$\mathsf{Enc}(|1\rangle) = \frac{1}{2\sqrt{2}}(|1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle + |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle), \quad (20)$$

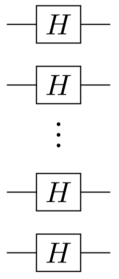
and the encoding of an arbitrary one-qubit pure state $|\psi\rangle$ and one-qubit mixed state ρ are defined by linearity.

We define the concatenated Steane codes by letting $\mathrm{Steane}_1 = \mathrm{Steane}$ denote the $[[7, 1, 3]]$ Steane code, and setting $\mathrm{Steane}_K = \mathrm{Steane}_{K-1} \circ \mathrm{Steane}$, a $[[7^K, 1, 3^K]]$ code. We notice that the concatenated Steane code can also be expressed in the stabilizer formalism. Suppose we have constructed a stabilizer code $\mathcal{S}_j$ for Steane_j for all $1 \leq j \leq K - 1$. Let $g_1, \ldots, g_6 \in \mathcal{S}_1$ be the stabilizers for Steane, and let $h_1, \ldots, h_{m-1}$ be a minimal generating set for $\mathcal{S}_{K-1}$, where $m = 7^{K-1}$. To encode the m physical qubits of Steane_{K-1} using Steane, we form the stabilizer $\mathcal{S}_1^{\otimes m}$. Define a homomorphism $\overline{\Delta} : \mathcal{P}_m \to \mathcal{P}_{7m}$ by sending $\Delta^i_{1,m}(W) \mapsto \Delta^i_{7,7m}(WWWWWWW)$ for each $W \in \mathcal{P}_1$ (in other words, we replace W in the ith position with $WWWWWWW$, so Z gets replaced with $\overline{Z}$, etc.). The operator $\overline{\Delta}(w)$ gives an encoding of w in Steane, so we can express Steane_K in the stabilizer formalism by taking the stabilizer code $\mathcal{S}_K$ with minimal generating set

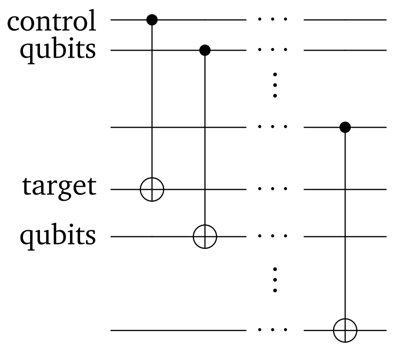
$$\{\Delta^i_{7,7m}(g_j), 1 \leq i \leq m, 1 \leq i \leq 6\} \cup \{\overline{\Delta}(h_j) : 1 \leq j \leq m - 1\}.$$

Steane_K contains stabilizers of weight 4 for all $K \geq 1$, so while Steane is nondegenerate, Steane_K is degenerate for $K \geq 2$.

Both H and $\Lambda(X)$ have transversal encodings in Steane_K, that is, encodings where H (resp. $\Lambda(X)$) is applied to each physical qubit (resp. each pair of physical qubits). More formally, $H(a)$ (the Hadamard gate acting on the ath logical qubit) is encoded by a sequence of gates $O_1(a), \ldots, O_\ell(a)$, where $\ell = 7^K$, and $O_i(a)$ is the Hadamard gate acting on the ith physical qubit of the ath logical qubit. The gate $\Lambda(X)(\underline{a})$ is encoded as a sequence of gates $O_1(\underline{a}), \ldots, O_\ell(\underline{a})$, where $\ell = 7^K$, and $O_i(a_1, a_2)$ is the $\Lambda(X)$ gate with the control on the ith physical qubit of the a_1th logical qubit, and the X gate on the ith physical qubit of the a_2th logical qubit. In both cases, no ancilla states are necessary. These encodings are shown in Figure 9.



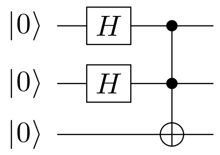
(a) H on encoded single-qubit state



(b) $\Lambda(X)$ on encoded two-qubit state

Figure 9: Steane_K encodings of H and $\Lambda(X)$

In order to complete a universal gateset, we show now how to apply Toffoli gates using Clifford operations and Toffoli magic states. The Toffoli magic state is the three-qubit state prepared by the following circuit:



Explicitly, the state is

$$|\text{Toffoli}\rangle = \frac{1}{2}\sum_{d,e} |d, e, de\rangle.$$

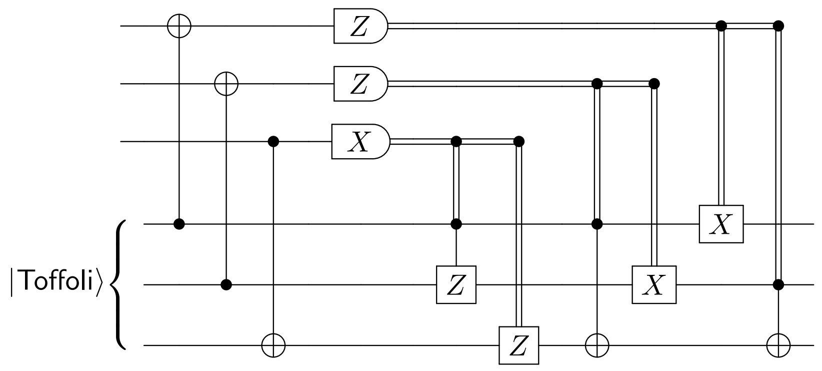


Figure 10: The Toffoli gadget

The Toffoli gate can be implemented using the Toffoli magic state, measurements of Z and X observables, and classically controlled Clifford gates, using the circuit shown in Figure 10 (the double wires in this circuit are classical bits).

5.5.2 Concatenated Steane code is simulatable

We are now ready to prove Theorem 6.

Proof of Theorem 6. Let $K > \log(s+3)$, and let us consider the Steane_K code. We have that it is is a $[[7^K, 1, 3^K]]$ stabilizer code, and since s is constant, so is K and $m = 7^K$.

As discussed previously, the gates H and $\Lambda(X)$ can be implemented transversally in Steane_K, and the Toffoli gates can be applied using $|\text{Toffoli}\rangle$ magic-state and the circuit in Figure 10. We remark that in such a gadget, we need to apply controlled H and $\Lambda(X)$ gates, and Steane_K is order-consistent with such gates. Finally, it is not hard to calculate that the measurement result of Figure 10 are uniformly random bits for all input states ρ. In this case, all of the assumptions of Propositions 25 and 29 are attained, and therefore Steane_K is a s-simulatable code. $\square$

A Differences between [19] and the zero knowledge protocol V_{ZK}

The structure and format of the protocol V_{ZK} is essentially the same as the protocols that arise from protocol compression in [19]. However, we list the few differences and provide explanations for why the soundness of the protocol is unaffected by these changes.

1. The outer code for V_{ZK} is a 4-qubit error detecting code, instead of the 7-qubit Steane code. As mentioned in Section 2, the soundness analysis of [19] only requires two properties from the outer code, and those properties are satisfied by the 4-qubit error detecting code. This is why we are able to have fewer additional provers than in the protocol compression result of [19].

2. The questions to the verifier players in V_{ZK} are six tuples of commuting two-qubit Pauli observables, whereas in [19] they are triples. This is because the verifier players are in charge of measuring the clock qubits as well as the snapshot qubits, whereas in [19] the clock measurements were delegated to a different set of players. However the clock measurements are also just Pauli measurements, so we can simply merge the snapshot and clock measurements together.

3. In the protocol V_{ZK}, the referee may ask questions QF_i or AF_i to prover player PP_i, which does not occur in the compression protocol of [19]. The referee will ask these questions when it decides to check the propagation of the gate at time $t_\star(i) - 1$ (in which case it will send question QF_i to PP_i), or at time $t_\star(i) + 1$ (in which case it will send question AF_i to PP_i). The check performed by the referee is the identical to that when it tests the propagation of the prover gate $\star_i$.

   For completeness, in the honest strategy the prover player PP_i measures a σ_X on a designated "question flag" register (when asked question QF_i), or measures a σ_X on a designated "answer flag" register (when asked question AF_i).

   Soundness is unaffected. If there was no valid history state before the addition of the QF and AF questions, then there is no valid history state with them.

References

[1] Dorit Aharonov. A simple proof that Toffoli and Hadamard are quantum universal. *CoRR*, abs/0301040, 2003.

[2] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[3] Mohammad Bavarian, Thomas Vidick, and Henry Yuen. Hardness amplification for entangled games via anchoring. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 303–316, 2017.

[4] John Stewart Bell. On the Einstein Podolsky Rosen paradox. *Physics*, 1(3):195–200, 1964.

[5] Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does parallel repetition lower the error in computationally sound protocols? In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 374–383, 1997.

[6] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 113–131, 1988.

[7] Anne Broadbent, Zhengfeng Ji, Fang Song, and John Watrous. Zero-knowledge proof systems for QMA. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 31–40, 2016.

[8] André Chailloux and Iordanis Kerenidis. Increasing the power of the verifier in quantum zero knowledge. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2008, December 9-11, 2008, Bangalore, India*, pages 95–106, 2008.

[9] Rui Chao, Ben W. Reichardt, Chris Sutherland, and Thomas Vidick. Test for a large amount of entanglement, using few measurements. *Quantum*, 2:92, September 2018.

[10] Alessandro Chiesa, Michael A. Forbes, Tom Gur, and Nicholas Spooner. Spatial isolation implies zero knowledge even in a quantum world. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 755–765, 2018.

[11] Richard Cleve, Peter Høyer, Benjamin Toner, and John Watrous. Consequences and limits of nonlocal strategies. In *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*, pages 236–249, 2004.

[12] Andrea Coladangelo. Parallel self-testing of (tilted) EPR pairs via copies of (tilted) CHSH and the magic square game. *Quantum Information & Computation*, 17(9&10):831–865, 2017.

[13] Andrea Coladangelo, Alex Bredariol Grilo, Stacey Jeffery, and Thomas Vidick. Verifier-on-a-leash: new schemes for verifiable delegated quantum computation, with quasilinear resources. *IACR Cryptology ePrint Archive*, 2019:247, 2019.

[14] Stephen A Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, 1973.

[15] Matthew Coudron and Anand Natarajan. The Parallel-Repeated Magic Square Game is Rigid. *CoRR*, abs/1609.06306, 2016.

[16] Matthew Coudron and William Slofstra. Complexity lower bounds for computing the approximately-commuting operator value of non-local games to high precision. In preparation.

[17] Claude Crépeau and Nan Yang. Non-locality in interactive proofs. *arXiv preprint arXiv:1801.04598*, 2018.

[18] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, May 1935.

[19] Joseph F. Fitzsimons, Zhengfeng Ji, Thomas Vidick, and Henry Yuen. Quantum proof systems for iterated exponential time, and beyond. *CoRR*, abs/1805.12166, 2018.

[20] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996.

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

[22] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, 1997.

[23] Markus Grassl, Th Beth, and Thomas Pellizzari. Codes for the quantum erasure channel. *Physical Review A*, 56(1):33, 1997.

[24] Tsuyoshi Ito and Thomas Vidick. A multi-prover interactive proof for NEXP sound against entangled provers. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 243–252, 2012.

[25] Zhengfeng Ji. Compression of quantum multi-prover interactive proofs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 289–302, 2017.

[26] Yusuke Kinoshita. Analysis of quantum multi-prover zero-knowledge systems: Elimination of the honest condition and computational zero-knowledge systems for QMIP. *CoRR*, abs/1902.10851, 2019.

[27] A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, Boston, MA, USA, 2002.

[28] Matthew McKague. Self-testing in parallel with CHSH. *Quantum*, 1:1, April 2017.

[29] Anand Natarajan and Thomas Vidick. A quantum linearity test for robustly verifying entanglement. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1003–1015, 2017.

[30] Anand Natarajan and Thomas Vidick. Low-degree testing for quantum states, and a quantum entangled games PCP for QMA. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 731–742, 2018.

[31] Anand Natarajan and Thomas Vidick. Two-player entangled games are np-hard. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 20:1–20:18, 2018.

[32] Anand Natarajan and John Wright. NEEXP in MIP*. *arXiv preprint arXiv:1904.05870*, 2019.

[33] Rafael Pass. Parallel repetition of zero-knowledge proofs and the possibility of basing cryptography on np-hardness. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 96–110, 2006.

[34] Ran Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.

[35] Ben W. Reichardt, Falk Unger, and Umesh V. Vazirani. Classical command of quantum systems. *Nature*, 496(7446):456–460, 2013.

[36] William Slofstra. Tsirelson's problem and an embedding theorem for groups arising from non-local games. *arXiv preprint arXiv:1606.03140*, 2016.

[37] William Slofstra. The set of quantum correlations is not closed. In *Forum of Mathematics, Pi*, volume 7. Cambridge University Press, 2019.

[38] Thomas Vidick, John Watrous, et al. Quantum proofs. *Foundations and Trends® in Theoretical Computer Science*, 11(1-2):1–215, 2016.

[39] John Watrous. Limits on the power of quantum statistical zero-knowledge. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, page 459, 2002.

[40] Henry Yuen. A parallel repetition theorem for all entangled games. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 77:1–77:13, 2016.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il