Secure Multi-party Quantum Computation with a Dishonest Majority

Yfke Dulek[1], Alex Grilo[2], Stacey Jeffery[2], Christian Majenz[2], and Christian Schaffner[1]

[1]QuSoft and U. Amsterdam
[2]QuSoft and CWI, Amsterdam

Abstract

The cryptographic task of secure multi-party (classical) computation has received a lot of attention in the last decades. Even in the extreme case where a computation is performed between k mutually distrustful players, and security is required even for the single honest player if all other players are colluding adversaries, secure protocols are known. For quantum computation, on the other hand, protocols allowing arbitrary dishonest majority have only been proven for $k = 2$. In this work, we generalize the approach taken by Dupuis, Nielsen and Salvail (CRYPTO 2012) in the two-party setting to devise a secure, efficient protocol for multi-party quantum computation for any number of players k, and prove security against up to $k - 1$ colluding adversaries. The quantum round complexity of the protocol for computing a quantum circuit with g gates acting on w qubits is $O((w + g)k)$. To achieve efficiency, we develop a novel public verification protocol for the Clifford authentication code, and a testing protocol for magic-state inputs, both using classical multi-party computation.

1 Introduction

In secure multi-party computation (MPC), two or more players want to jointly compute some publicly known function on their private data, without revealing their inputs to the other players. Since its introduction by Yao [Yao82], MPC has been extensively developed in different setups, leading to applications of both theoretical and practical interest (see, e.g., [CDN15] for a detailed overview).

With the emergence of quantum technologies, it becomes necessary to understand its consequences in the field of MPC. First, classical MPC protocols have to be secured against quantum attacks. But also, the increasing number of applications where quantum computational power is desired motivates protocols enabling multi-party *quantum* computation (MPQC) on the players' private (possibly quantum) data. In this work, we focus on the second task. Informally, we say a MPQC protocol is secure if the following two properties hold: 1. Dishonest players gain no information about the honest players' private inputs. 2. If the players do not abort the protocol, then at the end of the protocol they share a state corresponding to the correct computation applied to the inputs of honest players (those that follow the protocol) and some choice of inputs for the dishonest players.

MPQC was first studied by Crépeau, Gottesman and Smith [CGS02], who proposed a k-party protocol based on verifiable secret sharing that is information-theoretically secure, but requires the

assumption that at most $k/6$ players are dishonest. The fraction $k/6$ was subsequently improved to $< k/2$ [BOCG+06] which is optimal for secret-sharing-based protocols due to no-cloning. The case of a dishonest majority was thus far only considered for $k = 2$ parties, where one of the two players can be dishonest [DNS10, DNS12, KMW17][1]. These protocols are based on different cryptographic techniques, in particular quantum authentication codes in conjunction with classical MPC [DNS10, DNS12] and quantum-secure bit commitment and oblivious transfer [KMW17].

In this work, we propose the first secure MPQC protocol for any number k of players in the dishonest majority setting, i.e., the case with up to $k - 1$ colluding adversarial players.[2] We remark that our result achieves *composable security*, which is proven according to the standard ideal-vs.-real definition. Like the protocol of [DNS12], on which our protocol is built, our protocol assumes a classical MPC that is secure against a dishonest majority, and achieves the same security guarantees as this classical MPC. In particular, if we instantiate this classical MPC with an MPC in the *pre-processing model* (see [BDOZ11, DPSZ12, KPR18, CDE+18]), our construction yields a MPQC protocol consisting of a classical "offline" phase used to produce authenticated shared randomness among the players, and a second "computation" phase, consisting of our protocol, combined with the "computation" phase of the classical MPC. The security of the "offline" phase requires computational assumptions, but assuming no attack was successful in this phase, the second phase has information theoretic security.

1.1 Prior work

Our protocol builds on the two-party protocol of Dupuis, Nielsen, and Salvail [DNS12], which we now describe in brief. The protocol uses a classical MPC protocol, and involves two parties, Alice and Bob, of whom at least one is honestly following the protocol. Alice and Bob encode their inputs using a technique called *swaddling*: if Alice has an input qubit $|\psi\rangle$, she first encodes it using the n-qubit Clifford code (see Definition 2.5), resulting in $A(|0^n\rangle \otimes |\psi\rangle)$, for some random $(n + 1)$-qubit Clifford A sampled by Alice, where n is the security parameter. Then, she sends the state to Bob, who puts another encoding on top of Alice's: he creates the "swaddled" state $B(A(|0^n\rangle \otimes |\psi\rangle) \otimes |0^n\rangle)$ for some random $(2n + 1)$-qubit Clifford B sampled by Bob. This encoded state consists of $2n + 1$ qubits, and the data qubit $|\psi\rangle$ sits in the middle.

If Bob wants to test the state at some point during the protocol, he simply needs to undo the Clifford B, and test that the last n qubits (called traps) are $|0\rangle$. However, if Alice wants to test the state, she needs to work together with Bob to access her traps. Using classical multi-party computation, they jointly sample a random $(n + 1)$-qubit Clifford B' which is only revealed to Bob, and compute a Clifford $T := (\mathbb{I}^{\otimes n} \otimes B')(A^\dagger \otimes \mathbb{I}^{\otimes n})B^\dagger$ that is only revealed to Alice. Alice, who will not learn any relevant information about B or B', can use T to "flip" the swaddle, revealing her n trap qubits for measurement. After checking that the first n qubits are $|0\rangle$, she adds a fresh $(2n + 1)$-qubit Clifford on top of the state to re-encode the state, before computation can continue.

Single-qubit Clifford gates are performed simply by classically updating the inner key: if a state is encrypted with Cliffords BA, updating the decryption key to $BAG^\dagger$ effectively applies the gate G. In order to avoid that the player holding the inner key B skips this step, both players keep

[1] In Kashefi and Pappa [KP17], they consider a non-symmetric setting where the protocol is secure only when some specific sets of $k - 1$ players are dishonest.

[2] In the case where there are k adversaries and no honest players, there is nobody whose input privacy and output authenticity is worth protecting.

track of their keys using a classical commitment scheme. This can be encapsulated in the classical MPC, which we can assume acts as a trusted third party with a memory.

CNOT operations and measurements are slightly more involved, and require both players to test the authenticity of the relevant states several times. Hence, the communication complexity scales linearly with the number of CNOTs and measurements in the circuit.

Finally, to perform T gates, the protocol makes use of so-called magic states. To obtain reliable magic states, Alice generates a large number of them, so that Bob can test a sufficiently large fraction. He decodes them (with Alice's help), and measures whether they are in the expected state. If all measurements succeeds, Bob can be sufficiently certain that the untested (but still encoded) magic states are in the correct state as well.

1.1.1 Extending two-party computation to multi-party computation

A natural question is the possibility of lifting a two-party computation protocol to a multi-party computation protocol in a natural way. We discuss some of the issues that arise from such an approach, making it either infeasible or inefficient.

Composing ideal functionalities. The first naive idea would be trying to split the k players in two groups and make the groups simulate the players of a two-party protocol, whereas internally, the players run $\frac{k}{2}$-party computation protocols for all steps in the two-party protocol. Those $\frac{k}{2}$-party protocols are in turn realized by running $\frac{k}{4}$-party protocols, et cetera, until at the lowest level, the players can run actual two-party protocols.

Trying to construct such a composition in a black-box way, using the *ideal functionality* of a two-party protocol, one immediately faces a problem: at the lower levels, players learn intermediate states of the circuit, because they receive plaintext outputs from the ideal two-party functionality. This would immediately break the privacy of the protocol. If, on the other hand, we require the ideal two-party functionality to output encoded states instead of plaintexts, then the size of the ciphertext will grow at each level. The overhead of this approach would be $O(n^{\log k})$, where $n \geqslant k$ is the security parameter of the encoding, which would make this overhead super-polynomial in the number of players.

Naive extension of DNS to multi-party. One could also try to extend [DNS12] to multiple parties by adapting the subprotocols to work for more than two players. While this approach would likely lead to a correct and secure protocol for k parties, the computational costs of such an extension could be high.

First, note that in such an extension, each party would need to append n trap qubits to the encoding of each qubit, causing an overhead in the ciphertext size that is linear in k. Secondly, in this naive extension, the players would need to create $\Theta(2^k)$ magic states for T gates (see Section 2.5), since each party would need to test at least half of the ones approved by all previous players.

1.2 Our contributions

Our protocol builds on the work of Dupuis, Nielsen, and Salvail [DNS10, DNS12], and like it, assumes a classical MPC, and achieves the same security guarantees as this classical MPC. In contrast to a naive extension of [DNS12], requiring $\Theta(2^k)$ magic states, the complexity of our protocol,

when considering a quantum circuit that contains, among other gates, g gates in $\{\mathsf{CNOT}, \mathsf{T}\}$ and acts on w qubits, scales as $O((g+w)k)$.

In order to efficiently extend the two-party protocol of [DNS12] to a general k-party protocol, we make two major alterations to the protocol:

Public authentication test. In [DNS12], given a security parameter n, each party adds n qubits in the state $|0\rangle$ to each input qubit in order to authenticate it. The size of each ciphertext is thus $2n+1$. The extra qubits serve as check qubits (or "traps") for each party, which can be measured at regular intervals: if they are non-zero, somebody tampered with the state.

In a straightforward generalization to k parties, the ciphertext size would become $kn+1$ per input qubit, putting a strain on the computing space of each player. In our protocol, the ciphertext size is constant in the number of players: it is usually $n+1$ per input qubit, temporarily increasing to $2n+1$ for qubits that are involved in a computation step. As an additional advantage, our protocol does not require that all players measure their traps every time a state needs to be checked for its authenticity.

To achieve this smaller ciphertext size, we introduce a *public authentication test*. Our protocol uses a single, shared set of traps for each qubit. If the protocol calls for the authentication to be checked, the player that currently holds the state cannot be trusted to simply measure those traps. Instead, she temporarily adds extra trap qubits, and fills them with an encrypted version of the content of the existing traps. Now she measures only the newly created ones. The encryption ensures that the measuring player does not know the expected measurement outcome. If she is dishonest and has tampered with the state, she would have to guess a random n-bit string, or be detected by the other players. We design a similar test that checks whether a player has honestly created the first set of traps for their input at encoding time.

Efficient magic-state preparation. For the computation of non-Clifford gates, the [DNS12] protocol requires the existence of authenticated "magic states", auxiliary qubits in a known and fixed state that aid in the computation. In a two-party setting, one of the players can create a large number of such states, and the other player can, if he distrusts the first player, test a random subset of them to check if they were honestly initialized. Those tested states are discarded, and the remaining states are used in the computation.

In a k-party setting, such a "cut-and-choose" strategy where all players want to test a sufficient number of states would require the first party to prepare an exponential number (in k) of authenticated magic states, which quickly gets infeasible as the number of players grows. Instead, we need a testing strategy where dishonest players have no control over which states are selected for testing. We ask the first player to create a polynomial number of authenticated magic states. Subsequently, we use classical MPC to sample random, disjoint subsets of the proposed magic states, one for each player. Each player continues to decrypt and test their subset of states. The random selection process implies that, conditioned on the test of the honest player(s) being successful, the remaining registers indeed contain encrypted states that are reasonably close to magic states. Finally, we use standard magic-state distillation to obtain auxiliary inputs that are exponentially close to magic states.

1.3 Overview of the protocol

We describe some details of the k-player quantum MPC protocol for circuits consisting of classically-controlled Clifford operations and measurements. Such circuits suffice to perform Clifford com-

putation and magic-state distillation, so that the protocol can be extended to arbitrary circuits using the technique described above. The protocol consists of several subprotocols, of which we highlight four here: input encoding, public authentication test, single-qubit gate application, and CNOT application. In the following description, the classical MPC is treated as a trusted third party with memory[3]. The general idea is to first ensure that initially all inputs are properly encoded into the Clifford authentication code, and to test the encoding after each computation step that exposes the encoded qubit to an attack. During the protocol, the encryption keys for the Clifford authentication code are only known to the MPC.

Input encoding. For an input qubit $|\psi\rangle$ of player i, the MPC hands each player a circuit for a random $(2n+1)$-qubit Clifford group element. Now player i appends $2n$ "trap" qubits initialized in the $|0\rangle$-state and applies her Clifford. The state is passed around, and all other players apply their Clifford one-by-one, resulting in a Clifford-encoded qubit $F(|\psi\rangle \left|0^{2n}\right\rangle)$ for which knowledge of the encoding key F is distributed among all players. The final step is our *public authentication test*, which is used in several of the other subprotocols as well. Its goal is to ensure that all players, including player i, have honestly followed the protocol.

The public authentication test (details). The player holding the state $F(|\psi\rangle \left|0^{2n}\right\rangle)$ (player i) will measure n out of the $2n$ trap qubits, which should all be 0. To enable player i to measure a random subset of n of the trap qubits, the MPC could instruct her to apply $(E \otimes \mathsf{X}^r)(\mathbb{I} \otimes U_\pi)F^\dagger$ to get $E(|\psi\rangle |0^n\rangle) \otimes |r\rangle$, where U_π permutes the $2n$ trap qubits by a random permutation π, and E is a random $(n+1)$ qubit Clifford, and $r \in \{0,1\}^n$ is a random string. Then when player i measures the last n trap qubits, if the encoding was correct, she will obtain r and communicate this to the MPC. However, this only guarantees that the remaining traps are correct up to polynomial error.

To get a stronger guarantee, we replace the random permutation with an element from the sufficiently rich yet still efficiently samplable group of invertible transformations over $\mathbb{F}^{2n}$, $\mathrm{GL}(2n, \mathbb{F}_2)$. An element $g \in \mathrm{GL}(2n, \mathbb{F}_2)$ maybe be viewed as a unitary U_g acting on computational basis states as $U_g |x\rangle = |gx\rangle$ where $x \in \{0,1\}^{2n}$. In particular, $U_g \left|0^{2n}\right\rangle = \left|0^{2n}\right\rangle$, so if all traps are in the state $|0\rangle$, applying U_g does not change this, whereas for non-zero x, $U_g |x\rangle = |x'\rangle$ for a *random* $x' \in \{0,1\}^{2n}$. Thus the MPC instructs player i to apply $(E \otimes \mathsf{X}^r)(\mathbb{I} \otimes U_g)F^\dagger$ to the state $F(|\psi\rangle \left|0^{2n}\right\rangle)$, then measure the last n qubits and return the result, aborting if it is not r. Crucially, $(E \otimes \mathsf{X}^r)(\mathbb{I} \otimes U_g)F^\dagger$ is given as an element of the Clifford group, hiding the structure of the unitary and, more importantly, the values of r and g. So if player i is dishonest and holds a corrupted state, she can only pass the MPC's test by guessing r. If player i correctly returns r, we have the guarantee that the remaining state is a Clifford-authenticated qubit with n traps, $E(|\psi\rangle |0^n\rangle)$, up to exponentially small error.

Single-qubit Clifford gate application. As in [DNS12], this is done by simply updating encryption key held by the MPC: If a state is currently encrypted with a Clifford E, decrypting with a "wrong" key $EG^\dagger$ has the effect of applying G to the state.

CNOT application. Applying a CNOT gate to two qubits is slightly more complicated: as they are encrypted separately, we cannot just implement the CNOT via a key update like in the case of single qubit Clifford gates. Instead, we bring the two encoded qubits together, and then run a protocol that is similar to input encoding using the $(2n+2)$-qubit register as "input", but using $2n$ additional traps instead of just n, and skipping the final authentication-testing step. The joint state now has $4n+2$ qubits and is encrypted with some Clifford F only known to the MPC. Afterwards,

[3]The most common way to achieve classical MPC against dishonest majority is in the so called pre-processing model, as suggested by the SPDZ [BDOZ11] and MASCOT [KOS16] families of protocols. We believe that these protocols can be made post-quantum secure, but that is beyond the scope of this paper.

CNOT can be applied via a key update, similarly to single-qubit Cliffords. To split up the qubits again afterwards, the executing player applies $(E_1 \otimes E_2)F^\dagger$, where E_1 and E_2 are freshly sampled by the MPC. The two encoded qubits can then be tested separately using the public authentication test.

1.4 Open problems

Our results leave a number of exciting open problems to be addressed in future work. Firstly, the scope of this work was to provide a protocol that reduces the problem of MPQC to classical MPC in an information-theoretically secure way. Hence we obtain an information-theoretically secure MPQC protocol *in the preprocessing model*, leaving the post-quantum secure instantiation of the latter as an open problem.

Another class of open problems concerns applications of MPQC. Classically, MPC can be used to devise zero-knowledge proofs [IKOS09] and digital signature schemes [CDG+17]. Are there quantum generalizations of these applications?

An interesting open question concerning our protocol more specifically is whether the CNOT sub-protocol can be replaced by a different one that has round complexity independent of the total number of players, reducing the round complexity of the whole protocol to $O(g + kw)$, where g is the number of gates in the circuit and w the number of qubits involved in the computation. We also wonder if it is possible to develop more efficient protocols for narrower classes of quantum computation, instead of arbitrary (polynomial-size) quantum circuits.

Finally, it would be interesting to investigate whether the public authentication test we use can be leveraged in protocols for specific MPC-related tasks like oblivious transfer.

1.5 Outline

In Section 2, we outline the necessary preliminaries and tools we will make use of in our protocol. In Section 3, we give a precise definition of MPQC. In Section 4, we describe how players encode their inputs to setup for computation in our protocol. In Section 5 we describe our protocol for Clifford circuits, and finally, in Section 6, we show how to extend this to universal quantum circuits in Clifford+T.

Acknowledgments

We thank Frédéric Dupuis, Florian Speelman, and Serge Fehr for useful discussions. CM is supported by an NWO Veni Innovational Research Grant under project number VI.Veni.192.159. SJ is supported by an NWO WISE Fellowship, an NWO Veni Innovational Research Grant under project number 639.021.752, and QuantERA project QuantAlgo 680-91-03. SJ is a CIFAR Fellow in the Quantum Information Science Program. CS and CM were supported by a NWO VIDI grant (Project No. 639.022.519).

2 Preliminaries

2.1 Notation

We assume familiarity with standard notation in quantum computation, such as (pure and mixed) quantum states, the Pauli gates X and Z, the Clifford gates H and CNOT, the non-Clifford gate T, and measurements.

We work in the quantum circuit model, with circuits C composed of elementary unitary gates (of the set Clifford+T), plus computational basis measurements. We consider those measurement gates to be destructive, i.e., to destroy the post-measurement state immediately, and only a classical wire to remain. Since subsequent gates in the circuit can still classically control on those measured wires, this point of view is as general as keeping the post-measurement states around.

For two circuits C_1 and C_2, we write $C_2 \circ C_1$ for the circuit that consists of executing C_1, followed by C_2. Similarly, for two protocols Π_1 and Π_2, we write $\Pi_2 \diamond \Pi_1$ for the execution of Π_1, followed by the execution of Π_2.

We use capital letters for both quantum registers (M, R, S, $T, \dots$) and unitaries (A, B, U, V, $W, \dots$). We write $|R|$ for the dimension of the Hilbert space in a register R. The registers in which a certain quantum state exists, or on which some map acts, are written as gray superscripts, whenever it may be unclear otherwise. For example, a unitary U that acts on register A, applied to a state ρ in the registers AB, is written as $U^A \rho^{AB} U^\dagger$, where the registers $U^\dagger$ acts on can be determined by finding the matching U and reading the grey subscripts. Note that we do not explicitly write the operation $\mathbb{I}^B$ with which U is in tensor product. The gray superscripts are purely informational, and do not signify any mathematical operation. If we want to denote, for example, a partial trace of the state ρ^{AB}, we use the conventional notation ρ_A.

For an n-bit string $s = s_1 s_2 \cdots s_n$, define $U^s := U^{s_1} \otimes U^{s_2} \otimes \cdots \otimes U^{s_n}$. For an n-element permutation $\pi \in S_n$, define P_π to be the unitary that permutes n qubits according to π:

$$P_\pi \left|\psi_1\right\rangle \ldots \left|\psi_n\right\rangle = \left|\psi_{\pi(1)}\right\rangle \ldots \left|\psi_{\pi(n)}\right\rangle .$$

Write $[k]$ for the set $\{1, 2, \dots, k\}$.

For a projector Π, write $\overline{\Pi}$ for its complement $\mathbb{I} - \Pi$. Write $\tau^R := \mathbb{I}/|R|$ for the fully mixed state on the register R.

Write $GL(n, F)$ for the general linear group of degree n over a field F. We refer to the Galois field of two elements as $\mathbb{F}_2$, the n-qubit Pauli group as $\mathscr{P}_n$, and the n-qubit Clifford group as $\mathscr{C}_n$. Whenever a protocol mandates handing an element from one of these groups, or more generally, a unitary operation, to an agent, we mean that a description of the group element is given, e.g. in form of a normal-form circuit.

Finally, for a quantum operation that may take multiple rounds of inputs and outputs, for example an environment $\mathcal{E}$ interacting with a protocol Π, we write $\mathcal{E} \leftrightarrows \Pi$ for the final output of $\mathcal{E}$ after the entire interaction.

2.2 Classical multi-party computation

For multi-party computations where possibly more than half of the players are corrupted by the adversary, it is well known that one cannot achieve *fairness* which asks that either all parties receive the protocol output or nobody does. Cleve has shown [Cle86] that in the case of dishonest majority there cannot exist MPC protocols that provide fairness and guaranteed output delivery. In this

setting, we cannot prevent a dishonest player from simply aborting the protocol at any point, for example after having learned an unfavorable outcome of the protocol, before the honest player(s) have obtained their output(s). Hence, we have to settle for protocols allowing abort.

Over the last years, the most efficient protocols for classical multi-party computation with abort are in the so-called *pre-processing model*, as introduced by the SPDZ-family of protocols [BDOZ11, DPSZ12, KPR18, CDE+18][4]. These protocols consist of two phases: The first "offline" phase is executed independently of the inputs to the actual MPC and produces authenticated shared randomness among the players, for example in the form of authenticated multiplication triples [Bea92]. These triples are used in the second phase to run a very efficient secure-function-evaluation protocol which is *UC information-theoretically* secure against active corruptions of up to $k-1$ players, see, e.g., [CDN15, Section 8.5].

At this point, we are unaware of any formal analysis of the post-quantum security of these schemes. However, it should follow directly from Unruh's lifting theorem [Unr10] (asserting that classically secure protocols remain (statistically) secure in the quantum world) that the UC security of the second (online) phase can be lifted to the post-quantum setting. As for the pre-processing phase, there are two main types of protocols:

1. The SPDZ-family make use of the homomorphic properties of computationally-secure public-key encryption systems to generate the authenticated multiplication triples. The scheme in [DPSZ12] uses somewhat homomorphic encryption which is built from lattice assumptions and might already be post-quantum secure. In addition, the players provide non-interactive zero-knowledge proofs that they have performed these operations correctly. Those proofs are typically not post-quantum secure, but should be replaced by post-quantum secure variants like lattice-based zk-SNARGs [GMNO18] or zk-STARKs [BBHR18].

2. The authors of MASCOT (Multi-party Arithmetic Secure Computation with Oblivious Transfer) [KOS16] suggest a way to avoid the use of expensive public-key cryptography altogether and propose to use oblivious transfer (OT) and consistency checks to generate authenticated multiplication triples. A large number of OTs can be obtained from a few base OTs by OT extension-techniques such as [KOS15]. Those techniques are currently only proven in the classical random-oracle model (ROM), but can possibly be proven in the QROM as well. A post-quantum secure base OT can be obtained from lattices [PVW08]. Recent even more efficient MPC schemes [CDE+18] have followed this OT-based approach as well.

Establishing full post-quantum security of classical multi-party computation is outside the scope of this paper. For the purpose of this paper, we assume that such a post-quantum secure classical multi-party computation is available. According to the discussion above, it suffices that a preprocessing phase has been successfully run in order to have UC information-theoretically secure function evaluation (SFE). Unlike for general adversary structures [HMZ08], in our case of threshold adversaries, one can obtain (reactive) MPC from SFE by outputting shares of the overall state to the players and by asking the players to input them again in the next phase, see [CDN15, Section 5.3.1].

Throughout this paper, we will utilize the following ideal MPC functionality as a black box:

[4] We refer to [ACR18, Section 5] for a recent overview of other models of active corruptions that tolerate a dishonest majority of players, such as identifiable abort, covert security and public auditability.

Definition 2.1 (Ideal classical k-party stateful computation with abort)**.** *Let $f_1, ..., f_k$ and f_S be public classical deterministic functions on $k+2$ inputs. Let a string s represent the internal state of the ideal functionality. (The first time the ideal functionality is called, s is empty.) Let $A \subsetneq [k]$ be a set of corrupted players.*

1. *Every player $i \in [k]$ chooses an input x_i of appropriate size, and sends it (securely) to the trusted third party.*
2. *The trusted third party samples a bit string r uniformly at random.*
3. *The trusted third party computes $f_i(s, x_1, ..., x_k, r)$ for all $i \in [k] \cup \{S\}$.*
4. *For all $i \in A$, the trusted third party sends $f_i(s, x_1, ..., x_k, r)$ to player i.*
5. *All $i \in A$ respond with a bit b_i, which is 1 if they choose to abort, or 0 otherwise.*
6. *If $b_j = 0$ for all j, the trusted third party sends $f_i(s, x_1, ..., x_k, r)$ to the other players $i \in [k] \backslash A$ and stores $f_S(s, x_1, ..., x_k, r)$ in an internal state register (replacing s). Otherwise, he sends an* `abort` *message to those players.*

2.3 Pauli filter

In our protocol, we use a technique which alters a channel that would act jointly on registers A and B, so that its actions on A are replaced by a flag bit into a separate register. The flag is set to 0 if the actions on A belong to some set $\mathcal{P}$, or to 1 otherwise. This way, the new channel "filters" the allowed actions of A.

Definition 2.2 (Pauli filter)**.** *For registers A and B with $|B| > 0$, let U^{AB} be a unitary, and let $\mathcal{P} \subseteq \left(\{0,1\}^{\log|A|}\right)^2$ contain pairs of bit strings. The $\mathcal{P}$-filter of U on register A, denoted $\mathsf{PauliFilter}^A_{\mathcal{P}}(U)$, is the map $B \to BF$ (where F is some single-qubit flag register) that results from the following operations:*

1. *Initialize two separate registers A and A' in the state $|\Phi\rangle\langle\Phi|$, where $|\Phi\rangle := \left(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\right)^{\otimes \log|A|}$. Half of each pair is stored in A, the other in A'.*
2. *Run U on AB.*
3. *Measure AA' with the projective measurement $\{\Pi, \mathbb{I} - \Pi\}$ for*

$$\Pi := \sum_{(a,b)\in\mathcal{P}} \left(\mathsf{X}^a\mathsf{Z}^b\right)^A |\Phi\rangle\langle\Phi| \left(\mathsf{Z}^b\mathsf{X}^a\right).$$

If the outcome is Π, set the F register to $|0\rangle\langle 0|$. Otherwise, set it to $|1\rangle\langle 1|$.

The functionality of the Pauli filter becomes clear in the following lemma, which we prove in Appendix B by straightforward calculation:

Lemma 2.3. *For registers A and B with $|B| > 0$, let U^{AB} be a unitary, and let $\mathcal{P} \subseteq \left(\{0,1\}^{\log|A|}\right)^2$. Write $U = \sum_{x,z}(\mathsf{X}^x\mathsf{Z}^z)^A \otimes U^B_{x,z}$. Then $\mathsf{PauliFilter}^A_{\mathcal{P}}(U)$ equals the map*

$$(\cdot) \mapsto \sum_{(a,b)\in\mathcal{P}} U^B_{a,b}(\cdot)U^\dagger_{a,b} \otimes |0\rangle\langle 0|^F + \sum_{(a,b)\notin\mathcal{P}} U^B_{a,b}(\cdot)U^\dagger_{a,b} \otimes |1\rangle\langle 1|^F$$

A special case of the Pauli filter for $\mathcal{P} = \{(0^{\log|A|}, 0^{\log|A|})\}$ is due to Broadbent and Wainewright [BW16]. This choice of $\mathcal{P}$ represents only identity: the operation $\mathsf{PauliFilter}_{\mathcal{P}}$ filters out any components of U that do not act as identity on A. We will denote this type of filter with the name $\mathsf{IdFilter}$.

In this work, we will also use $\mathsf{XFilter}$, which only accepts components of U that act trivially on register A in the computational basis. It is defined by choosing $\mathcal{P} = \{0^{\log|A|}\} \times \{0,1\}^{\log|A|}$.

Finally, we note that the functionality of the Pauli filter given in Definition 2.2 can generalized, or weakened in a sense, by choosing a different state than $|\Phi\rangle\langle\Phi|$. In this work, we will use the $\mathsf{ZeroFilter}$, which initializes AA' in the state $|00\rangle^{\log|A|}$, and measures using the projector $\Pi = |00\rangle\langle 00|$. It filters U by allowing only those Pauli operations that leave the computational-zero state (but not necessarily any other computational-basis states) unaltered:

$$(\cdot) \mapsto U_0^B(\cdot)U_0^\dagger \otimes |0\rangle\langle 0|^F + \sum_{a\neq 0} U_a^B(\cdot)U_a^\dagger \otimes |1\rangle\langle 1|^F,$$

where we abbreviate $U_a := \sum_b U_{a,b}$. Note that for $\mathsf{ZeroFilter}$, the extra register A' can also be left out. For all the filters, we use (black) superscripts to denote the registers where the filtering is applied, e.g. $\mathsf{XFilter}^R(U)^{S\to SF}$ for U acting on RS denotes that R is filtered, and the resulting map hence still acts on S and produces flag F.

2.4 Clifford authentication code

The protocol presented in this paper will rely on quantum authentication. The players will encode their inputs using a quantum authentication code to prevent the other, potentially adversarial, players from making unauthorized alterations to their data. That way, they can ensure that the output of the computation is in the correct logical state.

A quantum authentication code transforms a quantum state (the *logical* state or *plaintext*) into a larger quantum state (the *physical* state or *ciphertext*) in a way that depends on a secret key. An adversarial party that has access to the ciphertext, but does not know the secret key, cannot alter the logical state without being detected at decoding time.

More formally, an authentication code consists of an encoding map $\mathsf{Enc}_k^{M\to MT}$ and a decoding map $\mathsf{Dec}_k^{MT\to M}$, for a secret key k, which we usually assume that the key is drawn uniformly at random from some key set $\mathcal{K}$. The message register M is expanded with an extra register T to accommodate for the fact that the ciphertext requires more space than the plaintext.

An authentication code is correct if $\mathsf{Dec}_k \circ \mathsf{Enc}_k = \mathbb{I}$. It is secure if the decoding map rejects (e.g., by replacing the output with a fixed reject symbol $\perp$) whenever an attacker tried to alter an encoded state:

Definition 2.4 (Security of authentication codes [DNS10])**.** *Let* $(\mathsf{Enc}_k^{M\to MT}, \mathsf{Dec}_k^{MT\to M})$ *be a quantum authentication scheme for* k *in a key set* $\mathcal{K}$*. The scheme is* ε*-secure if for all CPTP maps* $\mathcal{A}^{MTR}$ *acting on the ciphertext and a side-information register* R*, there exist CP maps* Λ_{acc} *and* Λ_{rej} *such that* $\Lambda_{\mathsf{acc}} + \Lambda_{\mathsf{rej}}$ *is trace-preserving, and for all* ρ^{MR}*:*

$$\left\| \mathop{\mathbb{E}}_{k\in\mathcal{K}} \left[\mathsf{Dec}_k\left(\mathcal{A}\left(\mathsf{Enc}_k\left(\rho\right)\right)\right)\right] \;-\; \left(\Lambda_{\mathsf{acc}}^R(\rho) + |\perp\rangle\langle\perp|^M \otimes \mathrm{Tr}_M\left[\Lambda_{\mathsf{rej}}^R\left(\rho\right)\right]\right) \right\|_1 \leqslant \varepsilon.$$

A fairly simple but powerful authentication code is the Clifford code:

Definition 2.5 (Clifford code [ABOE10])**.** *The n-qubit Clifford code is defined by a key set $\mathscr{C}_{n+1}$, and the encoding and decoding maps for a $C \in \mathscr{C}_{n+1}$:*

$$\mathsf{Enc}_C(\rho^M) := C(\rho^M \otimes |0^n\rangle\langle 0^n|^T)C^\dagger,$$

$$\mathsf{Dec}_C(\sigma^{MT}) := \langle 0^n|^T C^\dagger \sigma C |0^n\rangle + |\perp\rangle\langle\perp|^M \otimes \mathrm{Tr}_M \left[\sum_{x\neq 0^n} \langle x| C^\dagger \sigma C |x\rangle\right].$$

Note that, from the point of view of someone who does not know the Clifford key C, the encoding of the Clifford code looks like a Clifford twirl (see Appendix A) of the input state plus some trap states.

We prove the security of the Clifford code in Appendix C.

2.5 Universal gate sets

It is well known that if, in addition to Clifford gates, we are able to apply *any* non-Clifford gate G, then we are able to achieve universal quantum computation. In this work, we focus on the non-Clifford T gate (or $\pi/8$ gate).

In several contexts, however, applying non-Clifford gates are not straightforward for different reasons: common quantum error-correcting codes do not accept transversal implementation of non-Clifford gates, the non-Clifford gates do not commute with the quantum one-time pad and, more importantly in this work, neither with the Clifford encoding.

In order to concentrate the hardness of non-Clifford gates in an offline pre-processing phase, we can use techniques from computation by teleportation if we have so-called *magic states* of the form $|\mathsf{T}\rangle := \mathsf{T}|+\rangle$. Using a single copy of this state as a resource, we are able to implement a T gate using the circuit in Figure 1. The circuit only requires (classically controlled) Clifford gates.

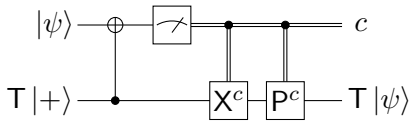

Figure 1: Using a magic state $|\mathsf{T}\rangle = \mathsf{T}|+\rangle$ to implement a T gate.

The problem is how to create such magic states in a fault-tolerant way. Bravyi and Kitaev [BK05] proposed a distillation protocol that allows to create states that are δ-close to true magic states, given $\mathrm{poly}(\log(1/\delta))$ copies of *noisy* magic-states. Let $\left|\mathsf{T}^\perp\right\rangle = \mathsf{T}|-\rangle$. Then we have:

Theorem 2.6 (Magic state distillation [BK05])**.** *There exists a circuit consisting of classically controlled Cliffords and computational-basis measurements such that for any $\varepsilon < \frac{1}{2}\left(1-\sqrt{3/7}\right)$, if ρ is the output on the first wire using input*

$$\left((1-\varepsilon)|\mathsf{T}\rangle\langle\mathsf{T}| + \varepsilon\left|\mathsf{T}^\perp\right\rangle\left\langle\mathsf{T}^\perp\right|\right)^{\otimes n}, \tag{1}$$

then $1 - \langle\mathsf{T}|\rho|\mathsf{T}\rangle \leqslant O\left((5\varepsilon)^{n^c}\right)$, where $c = (\log_2 30)^{-1} \approx 0.2$.

As we will see in Section 6, our starting point is a bit different from the input state required by Theorem 2.6. We now present a procedure that will allow us to prepare the states necessary for applying Theorem 2.6 (see Circuit 2.8). We prove Lemma 2.7 in Appendix D.

Lemma 2.7. *Let $V_{LW} = \mathrm{span}\{P_\pi(|\mathsf{T}\rangle^{\otimes m-w} \left|\mathsf{T}^\perp\right\rangle^w) : w \leqslant \ell, \pi \in S_m\}$, and let Π_{LW} be the orthogonal projector onto V_{LW}. Let Ξ denote the CPTP map induced by Circuit 2.8. If ρ is an m-qubit state such that $\mathrm{Tr}(\Pi_{LW}\rho) \geqslant 1 - \varepsilon$, then*

$$\left\| \Xi(\rho) - (|\mathsf{T}\rangle\langle\mathsf{T}|)^{\otimes t} \right\|_1 \leqslant O\left(m\sqrt{t} \left(\frac{\ell}{m}\right)^{O((m/t)^c/2)} + \varepsilon \right),$$

for some constant $c > 0$.

Circuit 2.8 (Magic-state distillation)**.** Given an m-qubit input state and a parameter $t < m$:

1. To each qubit, apply $\hat{\mathsf{Z}} := \mathsf{PX}$ with probability $\frac{1}{2}$.
2. Permute the qubits by a random $\pi \in S_m$.
3. Divide the m qubits into t blocks of size m/t, and apply magic-state distillation from Theorem 2.6 to each block.

Remark. Circuit 2.8 can be implemented with (classically controlled) Clifford gates and measurements in the computational basis.

3 Multi-party Quantum Computation: Definitions

In this section, we describe the ideal functionality we aim to achieve for multi-party quantum computation (MPQC) with a dishonest majority. As noted in Section 2.2, we cannot hope to achieve fairness: therefore, we consider an ideal functionality with the option for the dishonest players to abort.

Definition 3.1 (Ideal quantum k-party computation with abort)**.** *Let C be a quantum circuit on $W \in \mathbb{N}_{>0}$ wires. Consider a partition of the wires into the players' input registers plus an ancillary register, as $[W] = R_1^{\mathsf{in}} \sqcup \cdots \sqcup R_k^{\mathsf{in}} \sqcup R^{\mathsf{ancilla}}$, and a partition into the players' output registers plus a register that is discarded at the end of the computation, as $[W] = R_1^{\mathsf{out}} \sqcup \cdots \sqcup R_k^{\mathsf{out}} \sqcup R^{\mathsf{discard}}$. Let $I_A \subsetneq [k]$ be a set of corrupted players.*

1. *Every player $i \in [k]$ sends the content of R_i^{in} to the trusted third party.*
2. *The trusted third party populates R^{ancilla} with computational-zero states.*
3. *The trusted third party applies the quantum circuit C on the wires $[W]$.*
4. *For all $i \in I_A$, the trusted third party sends the content of R_i^{out} to player i.*
5. *All $i \in I_A$ respond with a bit b_i, which is 1 if they choose to abort, or 0 otherwise.*

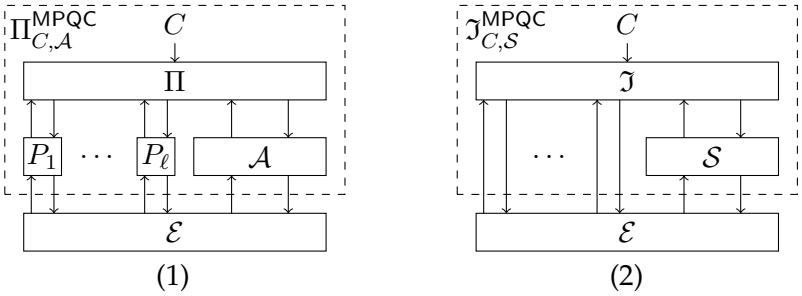

Figure 2: (1) The environment interacting with the protocol as run by honest players $P_1, \ldots, P_\ell$, and an adversary who has corrupted the remaining players. (2) The environment interacting with a simulator running the ideal functionality.

6. *If $b_i = 0$ for all i, the trusted third party sends the content of R_i^{out} to the other players $i \in [k] \backslash I_A$. Otherwise, he sends an* `abort` *message to those players.*

In Definition 3.1, all corrupted players individually choose whether to abort the protocol (and thereby to prevent the honest players from receiving their respective outputs). In reality, however, one cannot prevent several corrupted players from actively working together and sharing all information they have among each other. To ensure that our protocol is also secure in those scenarios, we consider security against a general adversary that corrupts all players in $I_{\mathcal{A}}$, by replacing their protocols by a single (interactive) algorithm $\mathcal{A}$ that receives the registers $R_{\mathcal{A}}^{\mathsf{in}} := R \sqcup \bigsqcup_{i \in I_{\mathcal{A}}} R_i^{\mathsf{in}}$ as input, and after the protocol produces output in the register $R_{\mathcal{A}}^{\mathsf{out}} := R \sqcup \bigsqcup_{i \in I_{\mathcal{A}}} R_i^{\mathsf{out}}$. Here, R is a side-information register in which the adversary may output extra information.

We will always consider protocols that fulfill the ideal functionality with respect to some gate set $\mathcal{G}$: the protocol should then mimic the ideal functionality only for circuits C that consist of gates from $\mathcal{G}$. This security is captured by the definition below.

Definition 3.2 (Computational security of quantum k-party computation with abort)**.** *Let $\mathcal{G}$ be a set of quantum gates. Let Π^{MPQC} be a k-party quantum computation protocol, parameterized by a security parameter n. For any circuit C, set $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players, and adversarial (interactive) algorithm $\mathcal{A}$ that performs all interactions of the players in $I_{\mathcal{A}}$, define $\Pi^{\mathsf{MPQC}}_{C,A} : R_{\mathcal{A}}^{\mathsf{in}} \sqcup \bigsqcup_{i \notin I_{\mathcal{A}}} R_i^{\mathsf{in}} \to R_{\mathcal{A}}^{\mathsf{out}} \sqcup \bigsqcup_{i \notin I_{\mathcal{A}}} R_i^{\mathsf{out}}$ to be the channel that executes the protocol Π^{MPQC} for circuit C by executing the honest interactions of the players in $[k] \setminus I_{\mathcal{A}}$, and letting $\mathcal{A}$ fulfill the role of the players in $I_{\mathcal{A}}$ (See Figure 2, (1)).*

For a simulator $\mathcal{S}$ that receives inputs in $R_{\mathcal{A}}^{\mathsf{in}}$, then interacts with the ideal functionalities on all interfaces for players in $I_{\mathcal{A}}$, and then produces output in $R_{\mathcal{A}}^{\mathsf{out}}$, let $\mathfrak{I}^{\mathsf{MPQC}}_{C,\mathcal{S}}$ be the ideal functionality described in Definition 3.1, for circuit C, simulator $\mathcal{S}$ for players $i \in I_{\mathcal{A}}$, and honest executions (with $b_i = 0$) for players $i \notin I_{\mathcal{A}}$ (See Figure 2, (2)). We say that Π^{MPQC} is a computationally ε-secure quantum k-party computation protocol with abort, if for all $I_{\mathcal{A}} \subsetneq [k]$, for all QPT adversaries $\mathcal{A}$, and all circuits C comprised of gates from $\mathcal{G}$, there exists a QPT simulator $\mathcal{S}$ such that for all QPT environments $\mathcal{E}$,

$$\left| \Pr\left[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi^{\mathsf{MPQC}}_{C,\mathcal{A}})\right] - \Pr\left[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}^{\mathsf{MPQC}}_{C,\mathcal{S}})\right] \right| \leqslant \varepsilon.$$

Here, the notation $b \leftarrow (\mathcal{E} \leftrightarrows (\cdot))$ represents the environment $\mathcal{E}$, on input 1^n, interacting with the (real or ideal) functionality $(\cdot)$, and producing a single bit b as output.

Remark. In the above definition, we assume that all QPT parties are polynomial in the size of circuit $|C|$, and in the security parameter n.

We show in Section 6.2 the protocol Π^{MPQC} implementing the ideal functionality described in Definition 3.1, and we prove its security in Theorem 6.5.

4 Setup and encoding

4.1 Input encoding

In the first phase of the protocol, all players encode their input registers qubit-by-qubit. For simplicity of presentation, we pretend that player 1 holds a single-qubit input state, and the other players do not have input. In the actual protocol, multiple players can hold multiple-qubit inputs: in that case, the initialization is run several times in parallel, using independent randomness. Any other player i can trivially take on the role of player 1 by relabeling the player indices.

Definition 4.1 (Ideal functionality for input encoding)**.** *Without loss of generality, let R_1^{in} be a single-qubit input register, and let $\dim(R_i^{\mathsf{in}}) = 0$ for all $i \neq 1$. Let $I_{\mathcal{A}} \subsetneq [k]$ be a set of corrupted players.*

1. *Player 1 sends register R_1^{in} to the trusted third party.*
2. *The trusted third party initializes a register T_1 with $|0^n\rangle\langle 0^n|$, applies a random $(n+1)$-qubit Clifford E to MT_1, and sends these registers to player 1.*
3. *All players $i \in I_{\mathcal{A}}$ send a bit b_i to the trusted third party. If $b_i = 0$ for all i, then the trusted third party stores the key E in the state register S of the ideal functionality. Otherwise, it aborts by storing $\perp$ in S.*

The following protocol implements the ideal functionality. It uses, as a black box, an ideal functionality MPC that implements a classical multi-party computation with memory.

Protocol 4.2. (Input encoding) Without loss of generality, let $M := R_1^{\mathsf{in}}$ be a single-qubit input register, and let $\dim(R_i^{\mathsf{in}}) = 0$ for all $i \neq 1$.

1. For every $i \in [k]$, MPC samples a random $(2n+1)$-qubit Clifford F_i and tells it to player i.
2. Player 1 applies the map $\rho^M \mapsto F_1 \left(\rho^M \otimes |0^{2n}\rangle\langle 0^{2n}|^{T_1T_2}\right) F_1^\dagger$ for two n-qubit (trap) registers T_1 and T_2, and sends the registers MT_1T_2 to player 2.
3. Every player $i = 2, 3, ..., k$ applies F_i to MT_1T_2, and forwards it to player $i+1$. Eventually, player k sends the registers back to player 1.
4. MPC samples a random $(n+1)$-qubit Clifford E, random n-bit string r and s, and a random classical invertible linear operator $g \in GL(2n, \mathbb{F})$. Let U_g be the (Clifford) unitary that computes g in-place, i.e., $U_g |t\rangle = |g(t)\rangle$ for all $t \in \{0,1\}^{2n}$.
5. MPC gives[a]
$$V := (E^{MT_1} \otimes (\mathsf{X}^r\mathsf{Z}^s)^{T_2})(\mathbb{I} \otimes (U_g)^{T_1T_2})(F_k \cdots F_2F_1)^\dagger$$

to player 1, who applies it to MT_1T_2.

6. Player 1 measures T_2 in the computational basis, discarding the measured wires, and keeps the other $(n+1)$ qubits as its output in $R_1^{\mathsf{out}} = MT_1$).

7. Player 1 submits the measurement outcome r' to MPC, who checks whether $r = r'$. If so, MPC stores the key E in its memory-state register S. If not, it aborts by storing $\perp$ in S.

[a] As described in Section 2.1, the MPC gives V as a group element, and the adversary cannot decompose it in the different parts that appear in its definition.

If MPC aborts the protocol in step 7, the information about the Clifford encoding key E is erased. In that case, the registers MT_1 will be fully mixed. Note that this result differs slightly from the 'reject' outcome of a quantum authentication code as in Definition 2.4, where the message register M is replaced by a dummy state $|\perp\rangle\langle\perp|$. In our current setting, the register M is in the hands of (the possibly malicious) player 1. We therefore cannot enforce the replacement of register M with a dummy state: we can only make sure that all its information content is removed. Depending on the application or setting, the trusted MPC can of course broadcast the fact that they aborted to all players, including the honest one(s).

To run Protocol 4.2 in parallel for multiple input qubits held by multiple players, MPC samples a list of Cliffords $F_{i,q}$ for each player $i \in [k]$ and each qubit q. The $F_{i,q}$ operations can be applied in parallel for all qubits q: with k rounds of communication, all qubits will have completed their round past all players.

We will show that Protocol 4.2 fulfills the ideal functionality for input encoding:

Lemma 4.3. *Let Π^{Enc} be Protocol 4.2, and $\mathfrak{I}^{\mathsf{Enc}}$ be the ideal functionality described in Definition 4.1. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$ with Π, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi_{\mathcal{A}}^{\mathsf{Enc}})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{\mathcal{S}}^{\mathsf{Enc}})| \leqslant \mathsf{negl}\,(n)\,.$$

Note that the environment $\mathcal{E}$ also receives the state register S, which acts as the "output" register of the ideal functionality (in the simulated case) or of MPC (in the real case). It is important that the environment cannot distinguish between the output states even given that state register S, because we want to be able to compose Protocol 5.4 with other protocols that use the key information inside S. In other words, it is important that, unless the key is discarded, the states *inside* the Clifford encoding are also indistinguishable for the environment.

We provide just a sketch of the proof for Lemma 4.3, and refer to Appendix E for its full proof.

Proof sketch. We divide our proof into two cases: when player 1 is honest, or when she is dishonest.

For the case when player 1 is honest, we know that she correctly prepares the expected state before the state is given to the other players. That is, she appends $2n$ ancilla qubits and applies the random Clifford instructed by the classical MPC. When the encoded state is returned to player 1, and she performs the Clifford V as instructed by the MPC, then by the properties of the Clifford encoding, the tested traps will be non-zero with probability exponentially close to 1, unless the other players performed the honest strategies.

The second case is a bit more complicated: the first player has full control over the state and, more importantly, the traps that will be used in the first encoding. In particular, she could start

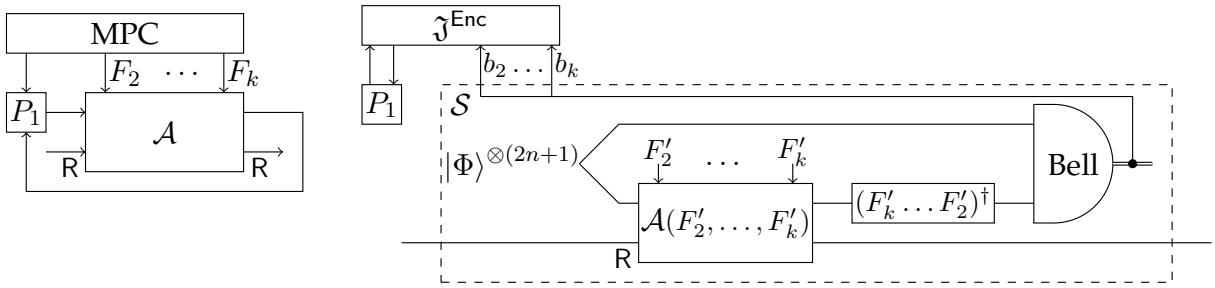

Figure 3: On the left, the adversary's interaction with the protocol Π^{Enc}, $\Pi^{\mathsf{Enc}}_{\mathcal{A}}$ in case player 1 is the only honest player. On the right, the simulator's interaction with $\mathfrak{J}^{\mathsf{Enc}}$, $\mathfrak{J}^{\mathsf{Enc}}_{\mathcal{S}}$. It performs the Pauli filter $\mathsf{IdFilter}^{MT_1T_2}$ on the adversary's attack on the encoded state.

with nonzero traps, which could possibly give some advantage to the dishonest players later on the execution of the protocol.

In order to prevent this type of attack, the MPC instructs the first player to apply a random linear function U_g on the traps, which is hidden from the players inside the Clifford V. If the traps were initially zero, their value does not change, but otherwise, they will be mapped to a random value, unknown by the dishonest parties. As such, the map U_g removes any advantage that the dishonest parties could have in step 7 by starting with non-zero traps. Because *any* nonzero trap state in T_1T_2 is mapped to a random string, it suffices to measure only T_2 in order to be certain that T_1 is also in the all-zero state (except with negligible probability). This intuition is formalized in Lemma E.1 in Appendix E.

Other possible attacks are dealt with in a way that is similar to the case where player 1 is honest (but from the perspective of another honest player).

In the full proof (see Appendix E), we present two simulators, one for each case, that tests (using Pauli filters from Section 2.3) whether the adversary performs any such attacks during the protocol, and chooses the input to the ideal functionality accordingly. See Figure 3 for a pictorial representation of the structure of the simulator for the case where player 1 is honest. □

4.2 Preparing ancilla qubits

Apart from encrypting the players' inputs, we also need a way to obtain encoded ancilla-zero states, which may be fed as additional input to the circuit. Since none of the players can be trusted to simply generate these states as part of their input, we need to treat them separately.

In [DNS10], Alice generates an encoding of $|0\rangle\langle 0|$, and Bob tests it by entangling (with the help of the classical MPC) the data qubit with a separate $|0\rangle\langle 0|$ qubit. Upon measuring that qubit, Bob then either detects a maliciously generated data qubit, or collapses it into the correct state. For details, see [DNS10, Appendix E].

Here, we take a similar approach, except with a public test on the shared traps. In order to guard against a player that may lie about the measurement outcomes during a test, we entangle the data qubits with *all* traps. We do so using a random linear operator, similarly to the encoding described in the previous subsection.

Essentially, the protocol for preparing ancilla qubits is identical to Protocol 4.2 for input encoding, except that now we do not only test whether the $2n$ traps are in the $|0\rangle\langle 0|$ state, but also the data qubit: concretely, the linear operator g acts on $2n+1$ elements instead of $2n$. That is,

$$V := (E \otimes P)U_g(F_k \cdots F_2 F_1)^\dagger.$$

As a convention, Player 1 will always create the ancilla $|0\rangle\langle 0|$ states and encode them. In principle, the ancillas can be created by any other player, or by all players together.

Per the same proof as for Lemma 4.3, we have implemented the following ideal functionality, again making use of a classical MPC as a black box.

Definition 4.4 (Ideal functionality for encoding of $|0\rangle\langle 0|$)**.** *Let $I_{\mathcal{A}} \subsetneq [k]$ be a set of corrupted players.*

1. *The trusted third party initializes a register T_1 with $|0^n\rangle\langle 0^n|$, applies a random $(n+1)$-qubit Clifford E to MT_1, and sends these registers to player 1.*
2. *All players $i \in I_{\mathcal{A}}$ send a bit b_i to the trusted third party. If $b_i = 0$ for all i, then the trusted third party stores the key E in the state register S of the ideal functionality. Otherwise, it aborts by storing $\perp$ in S.*

5 Computation of Clifford and measurement

After all players have successfully encoded their inputs and sufficiently many ancillary qubits, they perform a quantum computation gate-by-gate on their joint inputs. In this section, we will present a protocol for circuits that consist only of Clifford gates and computational-basis measurements. The Clifford gates may be classically controlled (for example, on the measurement outcomes that appear earlier in the circuit). In Section 6, we will discuss how to expand the protocol to general quantum circuits.

Concretely, we wish to achieve the functionality in Definition 3.1 for all circuits C that consist of Clifford gates and computational-basis measurements. As an intermediate step, we aim to achieve the following ideal functionality, where the players only receive an *encoded* output, for all such circuits:

Definition 5.1 (Ideal quantum k-party computation without decoding)**.** *Let C be a quantum circuit on W wires. Consider a partition of the wires into the players' input registers plus an ancillary register, as $[W] = R_1^{\mathsf{in}} \sqcup \cdots \sqcup R_k^{\mathsf{in}} \sqcup R^{\mathsf{ancilla}}$, and a partition into the players' output registers plus a register that is discarded at the end of the computation, as $[W] = R_1^{\mathsf{out}} \sqcup \cdots \sqcup R_k^{\mathsf{out}} \sqcup R^{\mathsf{discard}}$. Let $I_A \subsetneq [k]$ be the set of corrupted players.*

1. *All players i send their register R_i^{in} to the trusted third party.*
2. *The trusted third party instantiates R^{ancilla} with $|0\rangle\langle 0|$ states.*
3. *The trusted third party applies C to the wires $[W]$.*
4. *For every player i and every output wire $w \in R^{\mathsf{out}_i}$, the trusted third party samples a random $(n+1)$-qubit Clifford E_w, applies $\rho \mapsto E_w(\rho \otimes |0^n\rangle\langle 0^n|)E_w^\dagger$ to w, and sends the result to player i.*
5. *All players $i \in I_A$ send a bit b_i to the trusted third party.*

(a) If $b_i = 0$ for all i, all keys E_w and all measurement outcomes are stored in the state register S.

(b) Otherwise, the trusted third party `aborts` *by storing $\perp$ in S.*

To achieve the ideal functionality, we define several subprotocols. The subprotocols for encoding the players' inputs and ancillary qubits have already been described in Section 4. It remains to describe the subprotocols for (classically-controlled) single-qubit Clifford gates (Section 5.1), (classically controlled) CNOT gates (Section 5.2), and computational-basis measurements (Section 5.3).

In Section 5.5, we show how to combine the subprotocols in order to compute any polynomial-sized Clifford+measurement circuit. Our approach is inductive in the number of gates in the circuit. The base case is the identity circuit, which is essentially covered in Section 4. In Sections 5.1–5.3, we show that the ideal functionality for any circuit C, followed by the subprotocol for a gate G, results in the ideal functionality for the circuit $G \circ C$ (C followed by G). As such, we can chain together the subprotocols to realize the ideal functionality in Definition 5.1 for any polynomial-sized Clifford+measurement circuit. Combined with the decoding subprotocol we present in Section 5.4, such a chain of subprotocols satisfies Definition 3.1 for ideal k-party quantum Clifford+measurement computation with abort.

In Definition 5.1, all measurement outcomes are stored in the state register of the ideal functionality. We do so to ensure that the measurement results can be used as a classical control to gates that are applied after the circuit C, which can be technically required when building up to the ideal functionality for C inductively. Our protocols can easily be altered to broadcast measurement results as they happen, but the functionality presented in Definition 5.1 is the most general: if some player is supposed to learn a measurement outcome m_ℓ, then the circuit can contain a gate X^{m_ℓ} on an ancillary zero qubit that will be part of that player's output.

5.1 Subprotocol: single-qubit Cliffords

Due to the structure of the Clifford code, applying single-qubit Clifford is simple: the classical MPC, who keeps track of the encoding keys, can simply update the key so that it includes the single-qubit Clifford on the data register. We describe the case of a single-qubit Clifford that is classically controlled on a previous measurement outcome stored in the MPC's state. The unconditional case can be trivially obtained by omitting the conditioning.

Protocol 5.2 (Single-qubit Cliffords)**.** Let G^{m_ℓ} be a single-qubit Clifford to be applied on a wire w (held by a player i), conditioned on a measurement outcome m_ℓ. Initially, player i holds an encoding of the state on that wire, and the classical MPC holds the encoding key E.

1. MPC reads result m_ℓ from its state register S, and updates its internally stored key E to $E((G^{m_\ell})^\dagger \otimes \mathsf{I}^{\otimes n})$.

If $m_\ell = 0$, nothing happens. To see that the protocol is correct for $m_\ell = 1$, consider what happens if the state $E(\rho \otimes |0^n\rangle\langle 0^n|)E^\dagger$ is decoded using the updated key: the decoded output is

$$(E(G^\dagger \otimes \mathsf{I}^{\otimes n}))^\dagger E(\rho \otimes |0^n\rangle\langle 0^n|)E^\dagger(E(G^\dagger \otimes \mathsf{I}^{\otimes n})) \quad = \quad G\rho G^\dagger \otimes |0^n\rangle\langle 0^n|\,.$$

Protocol 5.2 implements the ideal functionality securely: given an ideal implementation $\mathfrak{I}^C$ for some circuit C, we can implement $G^{m_\ell} \circ C$ (i.e., the circuit C followed by the gate G^{m_ℓ}) by performing Protocol 5.2 right after the interaction with $\mathfrak{I}^C$.

Lemma 5.3. *Let G^{m_ℓ} be a single-qubit Clifford to be applied on a wire w (held by a player i), conditioned on a measurement outcome m_ℓ. Let $\Pi^{G^{m_\ell}}$ be Protocol 5.2 for the gate G^{m_ℓ}, and $\mathfrak{I}^C$ be the ideal functionality for a circuit C as described in Definition 5.1. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows (\Pi^{G^{m_\ell}} \diamond \mathfrak{I}^C)_{\mathcal{A}})] = \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{\mathcal{S}}^{G^{m_\ell} \circ C})].$$

Proof sketch. In the protocol $\Pi^{G^{m_\ell}} \diamond \mathfrak{I}^C$, an adversary has two opportunities to attack: once before its input state is submitted to $\mathfrak{I}^C$, and once afterwards. We define a simulator that applies these same attacks, except that it interacts with the ideal functionality $\mathfrak{I}^{G^{m_\ell} \circ C}$.

Syntactically, the state register S of $\mathfrak{I}^C$ is provided as input to the MPC in $\Pi^{G^{m_\ell}}$, so that the MPC can update the key as described by the protocol. As such, the output state of the adversary and the simulator are exactly equal. We provide a full proof in Appendix F. □

5.2 Subprotocol: CNOT gates

The application of two-qubit Clifford gates (such as CNOT) is more complicated than the single-qubit case, for two reasons.

First, a CNOT is a *joint* operation on two states that are encrypted with *separate* keys. If we were to classically update two keys E_1 and E_2 in a similar fashion as in Protocol 5.2, we would end up with a new key $(E_1 \otimes E_2)(\mathsf{CNOT}_{1,n+2})$, which cannot be written as a product of two separate keys. The keys would become 'entangled', which is undesirable for the rest of the computation.

Second, the input qubits might belong to separate players, who may not trust the authenticity of each other's qubits. In [DNS12], authenticity of the output state is guaranteed by having both players test each state several times. In a multi-party setting, both players involved in the CNOT are potentially dishonest, so it might seem necessary to involve all players in this extensive testing. However, because all our tests are publicly verified, our protocol requires less testing. Still, interaction with all other players is necessary to apply a fresh 'joint' Clifford on the two ciphertexts.

Protocol 5.4 (CNOT)**.** This protocol applies a CNOT gate to wires w_i (control) and w_j (target), conditioned on a measurement outcome m_ℓ. Suppose that player i holds an encoding of the first wire, in register $M^i T_1^i$, and player j of the second wire, in register $M^j T_1^j$. The classical MPC holds the encoding keys E_i and E_j.

1. If $i \neq j$, player j sends their registers $M^j T_1^j$ to player i. Player i now holds a $(2n+2)$-qubit state.
2. Player i initializes the registers T_2^i and T_2^j both in the state $|0^n\rangle\langle 0^n|$.
3. For all players h, MPC samples random $(4n+2)$-qubit Cliffords D_h, and gives them to the respective players. Starting with player i, each player h applies D_h to $M^{ij} T_{12}^{ij}$.[a] Eventually, player i receives the state back from player $i-1$. MPC remembers the applied Clifford
$$D := D_{i-1} D_{i-2} \cdots D_1 D_k D_{k-1} \cdots D_i.$$

4. MPC samples random $(2n+1)$-qubit Cliffords F_i and F_j, and tells player i to apply

$$V := (F_i \otimes F_j)\mathsf{CNOT}^{m_\ell}_{1,2n+2}(E_i^\dagger \otimes \mathsf{I}^{\otimes n} \otimes E_j^\dagger \otimes \mathsf{I}^{\otimes n})D^\dagger.$$

   Here, the CNOT acts on the two data qubits inside the encodings.

5. If $i \neq j$, player i sends $M^j T_{12}^j$ to player j.

6. Players i and j publicly tests their encodings. The procedures are identical, we describe the steps for player i:

   (a) MPC samples a random $(n+1)$-qubit Clifford E_i', which will be the new encoding key. Furthermore, MPC samples random n-bit strings s_i and r_i, and a random classical invertible linear operator g_i on $\mathbb{F}_2^{2n}$.

   (b) MPC tells player i to apply

$$W_i := (E_i' \otimes (\mathsf{X}^{r_i}\mathsf{Z}^{s_i})^{T_2^i})U_{g_i}^{T_{12}^i} F_i^\dagger.$$

   Here, U_{g_i} is as defined in Protocol 4.2.

   (c) Player i measures T_2^i in the computational basis and reports the n-bit measurement outcome r_i' to the MPC.

   (d) MPC checks whether $r_i' = r_i$. If it is not, MPC sends `abort` to all players. If it is, the test has passed, and MPC stores the new encoding key E_i' in their internal memory.

[a] We combine subscripts and superscripts to denote multiple registers: e.g., T_{12}^{ij} is shorthand for $T_1^i T_2^i T_1^j T_2^j$.

Lemma 5.5. *Let $\Pi^{\mathsf{CNOT}^{m_\ell}}$ be Protocol 5.4, to be executed on wires w_i and w_j, held by players i and j, respectively. Let $\mathfrak{I}^C$ be the ideal functionality for a circuit C as described in Definition 5.1. For all sets $I_\mathcal{A} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_\mathcal{A}$, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\left|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows (\Pi^{\mathsf{CNOT}^{m_\ell}} \diamond \mathfrak{I}^C)_\mathcal{A})] = \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_\mathcal{S}^{\mathsf{CNOT}^{m_\ell} \circ C})]\right| \leqslant \mathsf{negl}(n).$$

Proof sketch. There are four different cases, depending on which of players i and j are dishonest. In Appendix G, we provide a full proof by detailing the simulators for all four cases, but in this sketch, we only provide an intuition for the security in the case where both players are dishonest.

It is crucial that the adversary does not learn any information about the keys (E_i, E_j, E_i', E_j'), nor about the randomizing elements r_i, s_i, g_i. Even though the adversary learns W_i, W_j, and V explicitly during the protocol, all the secret information remains hidden by the randomizing Cliffords F_i, F_j, and D.

Consider a few ways in which the adversary may attack.

First, he may prepare a non-zero state in the registers T_2^i (or T_2^j) in step 2, potentially intending to spread those errors into $M^i T_1^i$ (or $M^j T_1^j$). Doing so, however, will cause U_{g_i} (or U_{g_j}) to map the trap state to a random non-zero string, and the adversary would not know what measurement string r_i' (or r_j') to report. Since g_i is unknown to the adversary, Lemma E.1 (see Appendix E) is applicable in this case: it states that it suffices to measure T_2^i in order to detect any errors in T_{12}^i.

Second, the adversary may fail to execute its instructions V or $W_i \otimes W_j$ correctly. Doing so is equivalent to attacking the state right before or right after these instructions. In both cases, however, the state in $M^i T_1^i$ is Clifford-encoded (and the state in T_2^i is Pauli-encoded) with keys unknown to the adversary, so the authentication property of the Clifford code prevents the adversary from altering the outcome.

The simulator we define in Appendix G tests the adversary exactly for the types of attacks above. By using Pauli filters (see Definition 2.2), the simulator checks whether the attacker leaves the authenticated states and the trap states T_2^i and T_2^j (both at initialization and before measurement) unaltered. In the full proof, we show that the output state of the simulator approximates, up to an error negligible in n, the output state of the real protocol. □

5.3 Subprotocol: Measurement

Measurement of authenticated states introduces a new conceptual challenge. For a random key E, the result of measuring $E(\rho \otimes |0^n\rangle\langle 0^n|)E^\dagger$ in a fixed basis is in no way correlated with the logical measurement outcome of the state ρ. However, the measuring player is also not allowed to learn the key E, so they cannot perform a measurement in a basis that depends meaningfully on E.

In [DNS10, Appendix E], this challenge is solved by entangling the state with an ancilla-zero state on a logical level. After this entanglement step, Alice gets the original state while Bob gets the ancilla state. They both decode their state (learning the key from the MPC), and can measure it. Because those states are entangled, and at least one of Alice and Bob is honest, they can ensure that the measurement outcome was not altered, simply by checking that they both obtained the same outcome. The same strategy can in principle also be scaled up to k players, by making all k players hold part of a big (logically) entangled state. However, doing so requires the application of $k-1$ logical CNOT operations, making it a relatively expensive procedure.

We take a different approach in our protocol. The player that performs the measurement essentially entangles, with the help of the MPC, the data qubit with a random subset of the traps. The MPC later checks the consistency of the outcomes: all entangled qubits should yield the same measurement result.

Our alternative approach has the additional benefit that the measurement outcome can be kept secret from some or all of the players. In the description of the protocol below, the MPC stores the measurement outcome in its internal state. This allows the MPC to classically control future gates on the outcome. If it is desired to instead reveal the outcome to one or more of the players, this can easily be done by performing a classically-controlled X operation on some unused output qubit of those players.

Protocol 5.6 (Computational-basis measurement)**.** Player i holds an encoding of the state in a wire w in the register MT_1. The classical MPC holds the encoding key E in the register S.

1. MPC samples random strings $r, s \in \{0,1\}^{n+1}$ and $c \in \{0,1\}^n$.
2. MPC tells player i to apply
$$V := \mathsf{X}^r \mathsf{Z}^s \mathsf{CNOT}_{1,c} E^\dagger$$

to the register MT_1, where $\mathsf{CNOT}_{1,c}$ denotes the unitary $\prod_{i\in[n]} \mathsf{CNOT}_{1,i}^{c_i}$ (that is, the string c dictates with which of the qubits in T_1 the M register will be entangled).

3. Player i measures the register MT_1 in the computational basis, reporting the result r' to MPC.

4. MPC checks whether $r' = r \oplus (m, m \cdot c)$ for some $m \in \{0,1\}$.[a] If so, it stores the measurement outcome m in the state register S. Otherwise, it aborts by storing $\perp$ in S.

5. MPC removes the key E from the state register S.

[a] The $\cdot$ symbol represents scalar multiplication of the bit m with the string c.

Lemma 5.7. *Let C be a circuit on W wires that leaves some wire $w \leqslant W$ unmeasured. Let $\mathfrak{I}^C$ the ideal functionality for C, as described in Definition 5.1, and let $\Pi^{\curlywedge}$ be Protocol 5.6 for a computational-basis measurement on w. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\left|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows (\Pi^{\curlywedge} \diamond \mathfrak{I}^C)_{\mathcal{A}})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{\mathcal{S}}^{\curlywedge \circ C})]\right| \leqslant \mathsf{negl}(n)\,.$$

Proof sketch. The operation $\mathsf{CNOT}_{1,c}$ entangles the data qubit in register M with a random subset of the trap qubits in register T_1, as dictated by c. In step 4 of Protocol 5.6, the MPC checks both for consistency of all the bits entangled by c (they have to match the measured data) *and* all the bits that are not entangled by c (they have to remain zero).

In Lemma H.1 in Appendix H, we show that checking the consistency of a measurement outcome after the application of $\mathsf{CNOT}_{1,c}$ is as good as measuring the logical state: any attacker that does not know c will have a hard time influencing the measurement outcome, as he will have to flip all qubits in positions i for which $c_i = 1$ without accidentally flipping any of the qubits in positions i for which $c_i = 0$. See Appendix H for a full proof that the output state in the real and simulated case are negligibly close. □

5.4 Subprotocol: Decoding

After the players run the computation subprotocols for all gates in the Clifford circuit, all they need to do is to decode their wires to recover their output. At this point, there is no need to check the authentication traps publicly: there is nothing to gain for a dishonest player by incorrectly measuring or lying about their measurement outcome. Hence, it is sufficient for all (honest) players to apply the regular decoding procedure for the Clifford code.

Below, we describe the decoding procedure for a single wire held by one of the players. If there are multiple output wires, then Protocol 5.8 can be run in parallel for all those wires.

Protocol 5.8 (Decoding)**.** Player i holds an encoding of the state w in the register MT_1. The classical MPC holds the encoding key E in the state register S.

1. MPC sends E to player i, removing it from the state register S.

2. Player i applies E to register MT_1.

3. Player i measures T_1 in the computational basis. If the outcome is not 0^n, player i discards M and aborts the protocol.

Lemma 5.9. *Let C be a circuit on W wires that leaves a single wire $w \leqslant W$ (intended for player i) unmeasured. Let $\mathfrak{I}^C$ be the ideal functionality for C, as described in Definition 5.1, and let $\mathfrak{I}_C^{\mathsf{MPQC}}$ be the ideal MPQC functionality for C, as described in Definition 3.1. Let Π^{Dec} be Protocol 5.8 for decoding wire w. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows (\Pi^{\mathsf{Dec}} \diamond \mathfrak{I}^C)_{\mathcal{A}})] = \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{C,\mathcal{S}}^{\mathsf{MPQC}})].$$

Proof sketch. If player i is honest, then he correctly decodes the state received from the ideal functionality $\mathfrak{I}^C$. A simulator would only have to compute the adversary's abort bit for $\mathfrak{I}_C^{\mathsf{MPQC}}$ based on whether the adversary decides to abort in either $\mathfrak{I}^C$ or the MPC computation in Π^{Dec}.

If player i is dishonest, a simulator $\mathcal{S}$ runs the adversary on the input state received from the environment before inputting the resulting state into the ideal functionality $\mathfrak{I}_C^{\mathsf{MPQC}}$. The simulator then samples a key for the Clifford code and encodes the output of $\mathfrak{I}_C^{\mathsf{MPQC}}$, before handing it back to the adversary. It then simulates Π^{Dec} by handing the sampled key to the adversary. If the adversary aborts in one of the two simulated protocols, then the simulator sends abort to the ideal functionality $\mathfrak{I}_C^{\mathsf{MPQC}}$. □

5.5 Combining Subprotocols

We show in this section how to combine the subprotocols of the previous sections in order to perform multi-party quantum Clifford computation.

Recalling the notation defined in Definition 3.1, let C be a quantum circuit on $W \in \mathbb{N}_{>0}$ wires, which are partitioned into the players' input registers plus an ancillary register, as $[W] = R_1^{\mathsf{in}} \sqcup \cdots \sqcup R_k^{\mathsf{in}} \sqcup R^{\mathsf{ancilla}}$, and a partition into the players' output registers plus a register that is discarded at the end of the computation, as $[W] = R_1^{\mathsf{out}} \sqcup \cdots \sqcup R_k^{\mathsf{out}} \sqcup R^{\mathsf{discard}}$. We assume that C is decomposed in a sequence $G_1, ..., G_m$ of operations where each G_i is one of the following operations:

- a single-qubit Clifford on some wire $j \in [M]$;
- a CNOT on wires $j_1, j_2 \in [M]$ for $j_1 \neq j_2$;
- a measurement of the qubit on wire j in the computational basis.

In Sections 4 and 5.1–5.3, we have presented subprotocols for encoding single qubits and perform these types of operations on single wires. The protocol for all players to jointly perform the bigger computation C is simply a concatenation of those smaller subprotocols:

Protocol 5.10 (Encoding and Clifford+measurement computation)**.** Let C be a Clifford + measurement circuit composed of the gates $G_1, \ldots, G_m$ on wires $[W]$ as described above.

1. For all $i \in [k]$ and $j \in R_i^{\mathsf{in}}$, run Protocol 4.2 for the qubit in wire j.
2. For all $j \in R^{\mathsf{ancilla}}$, run Protocol 4.2 (with the differences described in Section 4.2).

3. For all $j \in [m]$:
   (a) If G_j is a single-qubit Clifford, run Protocol 5.2 for G_j.
   (b) If G_j is a CNOT, run Protocol 5.4 for G_j.
   (c) If G_j is a computational-basis measurement, run Protocol 5.6 for G_j.
4. For all $i \in [k]$ and $j \in R_i^{\mathsf{out}}$, run Protocol 5.8 for the qubit in wire j.

Lemma 5.11. *Let Π^{Cliff} be Protocol 5.10, and $\mathfrak{I}^{\mathsf{Cliff}}$ be the ideal functionality described in Definition 3.1 for the special case where the circuit consists of Cliffords and measurements. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$ with Π, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi^{\mathsf{Cliff}}_{\mathcal{A}})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}^{\mathsf{Cliff}}_{\mathcal{S}})| \leqslant \mathsf{negl}(n).$$

Proof. First notice that $\mathfrak{I}^{\mathsf{Cliff}} = \mathfrak{I}^{G_m \circ \cdots \circ G_1 \circ \mathsf{Enc}}$ and $\Pi^{\mathsf{Cliff}} = \Pi^{\mathsf{Dec}} \diamond \Pi^{G_m} \diamond \ldots \diamond \Pi^{G_1} \diamond \Pi^{\mathsf{Enc}}$. For simplicity, for some circuit C' composed of gates $G'_1, \ldots G'_{m'}$, we denote $\Pi^{C'} = \Pi^{G'_{m'}} \diamond \ldots \diamond \Pi^{G'_1}$. We also denote $C'_{r,s}$, for $1 \leqslant r \leqslant s \leqslant m'$ as the circuit composed by gates $G'_r, \ldots, G'_s$.

We start by proving by induction that for all i, the following holds:

$$\left|\Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C} \diamond \Pi^{\mathsf{Enc}}_{\mathcal{A}})] - \Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i,m}} \diamond \mathfrak{I}^{C_{1,i-1} \circ \mathsf{Enc}}_{\mathcal{S}_{\mathsf{Enc}}})\right| \leqslant i \cdot \mathsf{negl}(n).$$

For the basis case $i = 1$, notice that from Lemma 4.3, there exists a simulator $\mathcal{S}_{\mathsf{Enc}}$ such that for all $\mathcal{E}'$

$$|\Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{\mathsf{Enc}}_{\mathcal{A}})] - \Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \mathfrak{I}^{\mathsf{Enc}}_{\mathcal{S}_{\mathsf{Enc}}})| \leqslant \mathsf{negl}(n),$$

therefore, in particular for every $\mathcal{E}''$ we have that

$$|\Pr[1 \leftarrow (\mathcal{E}'' \leftrightarrows \Pi^{C} \diamond \Pi^{\mathsf{Enc}}_{\mathcal{A}})] - \Pr[1 \leftarrow (\mathcal{E}'' \leftrightarrows \Pi^{C} \diamond \mathfrak{I}^{\mathsf{Enc}}_{\mathcal{S}_{\mathsf{Enc}}})| \leqslant \mathsf{negl}(n).$$

For the induction step, assume that our statement holds for some $i \geqslant 1$. Then if G_{i+1} is a single-qubit Clifford we have that

$$\begin{aligned}
&|\Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C} \diamond \Pi^{\mathsf{Enc}}_{\mathcal{A}})] - \Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i+1,m}} \diamond \mathfrak{I}^{C_{1,i} \circ \mathsf{Enc}}_{\mathcal{S}_{\mathsf{Enc}}})| \\
&\leqslant |\Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C} \diamond \Pi^{\mathsf{Enc}}_{\mathcal{A}})] - \Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i,m}} \diamond \mathfrak{I}^{C_{1,i-1} \circ \mathsf{Enc}}_{\mathcal{S}_{\mathsf{Enc}}})| \\
&+ |\Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i,m}} \diamond \mathfrak{I}^{C_{1,i-1} \circ \mathsf{Enc}}_{\mathcal{S}_{\mathsf{Enc}}}) - \Pr[1 \leftarrow (\mathcal{E}' \leftrightarrows \Pi^{C_{i+1,m}} \diamond \mathfrak{I}^{C_{1,i} \circ \mathsf{Enc}}_{\mathcal{S}_{\mathsf{Enc}}})| \\
&\leqslant (i+1)\mathsf{negl}(n),
\end{aligned}$$

where in the first step we use the triangle inequality and in the second step we use the induction hypothesis and Lemma 5.3.

For the cases where G_{i+1} is a CNOT or measurement, the same argument follows by using Lemmas 5.5 and 5.7 accordingly.

Finally, by Lemma 5.9, we can also replace Π^{Dec} by $\mathfrak{I}^{\mathsf{Dec}}$, at the cost of $\mathsf{negl}(n)$. As long as $m = \mathsf{poly}(n)$, the result follows since $m \cdot \mathsf{negl}(n) = \mathsf{negl}(n)$. □

6 Protocol: MPQC for general quantum circuits

In this section, we show how to lift the MPQC for Clifford operations (as laid out in Sections 4 and 5) to MPQC for general quantum circuits.

The main idea is to use magic states for T gates, as described in Section 2.5. Our main difficulty here is to that the magic states must be supplied by the possibly dishonest players themselves. We solve this problem in Section 6.1 and then in Section 6.2, we describe the MPQC protocol for universal computation combining the results from Sections 5 and 6.1.

6.1 Magic-state distillation

We now describe a subprotocol that allows the players to create the encoding of exponentially good magic states, if the players do not abort.

Our subprotocol can be divided into two parts. In the first part, player 1 is asked to create many magic states, which the other players will test. After this step, if none of the players abort during the testing, then with high probability the resource states created by player 1 are at least somewhat good. In the second part of the subprotocol, the players run a distillation procedure to further increase the quality of the magic states.

Protocol 6.1 (Magic-state creation)**.** Let t be the number of magic states we wish to create. Let $\ell := (t + k)n$.

1. Player 1 creates ℓ copies of $|\mathsf{T}\rangle$ and encodes them separately using Protocol 4.2 (jointly with the other players).
2. MPC picks disjoint sets $S_2, \ldots, S_k \subseteq [\ell]$ of size n each.
3. For each $i \in 2, \ldots, k$, player i decodes the magic states indicated by S_i (see Protocol 5.8), measures in the $\{|\mathsf{T}\rangle, |\mathsf{T}^\perp\rangle\}$-basis and aborts if any outcome is different from $|\mathsf{T}\rangle$.
4. On the remaining encoded states, the players run Protocol 5.10 for multi-party computation of Clifford circuits (but skipping the input-encoding step) to perform the magic-state distillation protocol describe in Protocol 2.8. Any randomness required in that protocol is sampled by the classical MPC.

We claim that Protocol 6.1 implements the following ideal functionality for creating t magic states, up to a negligible error:

Definition 6.2 (Ideal functionality for magic-state creation)**.** *Let t be the number of magic states we wish to create. Let $I_{\mathcal{A}} \subsetneq [k]$ be a set of corrupted players.*

1. *For every $i \in I_{\mathcal{A}}$, player i sends a bit b_i to the trusted third party.*
   (a) *If $b_i = 0$ for all i, the trusted third party samples t random $(n+1)$-qubit Clifford E_j for $1 \leqslant j \leqslant t$, and sends $E_j(|\mathsf{T}\rangle \otimes |0^n\rangle)$ to Player 1*
   (b) *Otherwise, the trusted third party sends* `abort` *to all players.*
2. *Store the keys E_j, for $1 \leqslant j \leqslant t$ in the state register S of the ideal functionality.*

Lemma 6.3. *Let Π^{MS} be Protocol 6.1, and $\mathfrak{I}^{MS}$ be the ideal functionality described in Definition 6.2. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$ with Π, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$\left|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi_{\mathcal{A}}^{MS})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{\mathcal{S}}^{MS})\right| \leqslant \mathsf{negl}\,(n)\,.$$

We prove this lemma in Appendix I.

6.2 MPQC protocol for universal quantum computation

Finally, we present our protocol for some arbitrary quantum computation. For this setting, we extend the setup of Section 5.5 by considering quantum circuits $C = G_m ... G_1$ where G_i can be single-qubit Cliffords, CNOTs, measurements or, additionally, T gates.

For that, we will consider a circuit C' where each gate $G_i = \mathsf{T}$ acting on qubit j is then replaced by the T-gadget presented in Figure 1, acting on the qubit j and a fresh new T magic states.

Protocol 6.4 (Protocol for universal MPQC)**.** Let C be a polynomial-sized quantum circuit, and t be the number of T-gates in C.

1. Run Protocol 6.1 to create t magic states.
2. Run Protocol 5.10 for the circuit C', which is equal to the circuit C, except each T gate is replaced with the T-gadget from Figure 1.

Theorem 6.5. *Let Π^{MPQC} be Protocol 6.4, and $\mathfrak{I}^{\mathsf{MPQC}}$ be the ideal functionality described in Definition 3.1. For all sets $I_{\mathcal{A}} \subsetneq [k]$ of corrupted players and all adversaries $\mathcal{A}$ that perform the interactions of players in $I_{\mathcal{A}}$ with Π, there exists a simulator $\mathcal{S}$ (the complexity of which scales polynomially in that of the adversary) such that for all environments $\mathcal{E}$,*

$$|\Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \Pi_{\mathcal{A}}^{\mathsf{MPQC}})] - \Pr[1 \leftarrow (\mathcal{E} \leftrightarrows \mathfrak{I}_{\mathcal{S}}^{\mathsf{MPQC}})| \leqslant \mathsf{negl}\,(n)\,.$$

Proof. Direct from Lemmas 5.11 and 6.3. □

References

[ABOE10] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computations. In *ICS 2010*, 2010.

[ABOEM17] Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive proofs for quantum computations. *arXiv preprint arXiv:1704.04487*, 2017.

[ACR18] Daniel Augot, André Chailloux, and Matthieu Rambaud. D3.5 cloud: Advanced applications. http://www.pqcrypto.eu.org/deliverables/D3.5cloud-final.pdf, 2018. Deliverable of H2020 project PQCRYPTO.

[AM17] Gorjan Alagic and Christian Majenz. Quantum non-malleability and authentication. In *CRYPTO 2017*, 2017.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT 2011*, 2011.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO 91*, 1992.

[BF10] Niek J. Bouman and Serge Fehr. Sampling in a quantum population, and applications. In *CRYPTO 2010*, 2010.

[BK05] Sergei Bravyi and Alexei Kitaev. Universal quantum computation with ideal clifford gates andnoisy ancillas. *Physical Review A*, (022316), 2005.

[BOCG+06] Michael Ben-Or, Claude Crépeau, Daniel Gottesman, Avinatan Hassidim, and Adam Smith. Secure multiparty quantum computation with (only) a strict honest majority. In *FOCS'06*, 2006.

[BW16] Anne Broadbent and Evelyn Wainewright. Efficient simulation for quantum message authentication. In *ICITS 2016*, 2016.

[CDE+18] Ronald Cramer, Ivan Damgrd, Daniel Escudero, Peter Scholl, and Chaoping Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority. *CRYPTO 2018*, 2018.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In *CCS '17*, 2017.

[CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[CGS02] Claude Crépeau, Daniel Gottesman, and Adam Smith. Secure multi-party quantum computation. In *STOC '02*, 2002.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC '86*, 1986.

[DNS10] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Secure two-party quantum evaluation of unitaries against specious adversaries. In *CRYPTO 2010*, 2010.

[DNS12] Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Actively secure two-party evaluation of any quantum operation. In *CRYPTO 2012*. 2012.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO 2012*, 2012.

[GMNO18] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-snarks from square span programs. In *CCS 2018*, 2018.

[HMZ08] Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE : Unconditional and computational security. In *ASIACRYPT 2008*, 2008.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3), 2009.

[KMW17] Elham Kashefi, Luka Music, and Petros Wallden. The quantum cut-and-choose technique and quantum two-party computation. *arXiv preprint arXiv:1703.03754*, 2017.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In *CRYPTO 2015*, 2015.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *CCS 2016*, 2016.

[KP17] Elham Kashefi and Anna Pappa. Multiparty delegated quantum computing. *Cryptography*, 1(2), 2017.

[KPR18] Marcel Keller, Valerio Pastro, and Dragos Rotaru. Overdrive: Making SPDZ great again. *EUROCRYPT 2018*, January 2018.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *CRYPTO 2008*, 2008.

[Unr10] Dominique Unruh. Universally composable quantum multi-party computation. In *EUROCRYPT 2010*, 2010.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS '82*, 1982.

Appendices

A Twirling

One of the techniques we use in this work is the twirl over a group $\mathcal{G}$ of unitary operators, which maps a state (or channel) to its '$\mathcal{G}$-averaged' version. Specifically, the twirl of a state ρ is defined

$$\mathcal{T}_{\mathcal{G}}(\rho) := \frac{1}{|\mathcal{G}|}\sum_{U\in\mathcal{G}} U\rho U^\dagger,$$

and the twirl of a channel Λ is defined as

$$\mathcal{T}_{\mathcal{G}}(\Lambda(\cdot)) := \frac{1}{|\mathcal{G}|}\sum_{U\in\mathcal{G}} U^\dagger(\Lambda(U(\cdot)U^\dagger))U.$$

We sometimes abuse notation for non-unitary groups: for example, throughout this work we use $\mathcal{T}_{GL(2n,\mathbb{F}_2)}(\cdot)$ to denote a twirl over the unitary group $\{U_g \mid g \in GL(2n,\mathbb{F}_2)\}$, where U_g is defined as the unitary that applies g in-place, i.e., $U_g|t\rangle = |g(t)\rangle$ for all $t \in \{0,1\}^{2n}$.

Twirling a state over the n-qubit Pauli group is equivalent to encrypting the state under the quantum one-time pad, and tracing out the encryption key. From the point of view of someone without that encryption key, the resulting state is fully mixed:

Lemma A.1 (Pauli twirl of a state)**.** *For all n-dimensional states ρ,*

$$\mathcal{T}_{\mathscr{P}_n}(\rho) = \tau.$$

Proof. Write $\rho = \sum_{i,j\in\{0,1\}^n} \alpha_{ij}|i\rangle\langle j|$. For every i,j we have

$$\begin{aligned}\mathcal{T}_{\mathscr{P}_n}(|i\rangle\langle j|) &= \mathop{\mathbb{E}}_{x,z\in\{0,1\}^n} \mathsf{X}^x\mathsf{Z}^z|i\rangle\langle j|\mathsf{Z}^z\mathsf{X}^x \\ &= \mathop{\mathbb{E}}_{x,z\in\{0,1\}^n} (-1)^{z(i\oplus j)}|i\oplus x\rangle\langle j\oplus x|.\end{aligned}$$

Note that $\mathbb{E}_{z\in\{0,1\}^n}(-1)^{z(i\oplus j)} = 0$ whenever $i \neq j$ (i.e., $i\oplus j \neq 0$), and that the term evaluates to 1 whenever $i = j$. So

$$\mathcal{T}_{\mathscr{P}_n}(|i\rangle\langle j|) = \begin{cases} \mathbb{E}_x|i\oplus x\rangle\langle i\oplus x| = \tau & \text{if } i = j \\ 0 & \text{otherwise.}\end{cases}$$

To conclude the proof, sum all terms of ρ to get

$$\mathcal{T}_{\mathscr{P}_n}(\rho) = \sum_{i,j}\alpha_{ij}\mathcal{T}_{\mathscr{P}_n}(|i\rangle\langle j|) = \sum_i \alpha_{ii}\tau = \tau.$$

□

Since the Pauli group is a subgroup of the Clifford group, Lemma A.1 also holds when twirling over the n-qubit Clifford group.

On a channel, the Pauli twirl has a similar effect of transforming a superposition of attack maps into a classical mixture of Pauli attacks. This transformation greatly simplifies analysis:

Lemma A.2 (Pauli twirl of a channel [ABOEM17, Lemma 5.1]). *For all* $V^{AB} = \sum_{P \in \mathscr{P}_n} P^A \otimes V_P^B$,

$$\mathcal{T}^A_{\mathscr{P}_n}(V(\cdot)V^\dagger) = \sum_{P \in \mathscr{P}_n} (P \otimes V_P)(\cdot)(P \otimes V_P)^\dagger,$$

where the twirl is applied on the 2^n*-dimensional register* A.

B Proof of Lemma 2.3

Proof. Given an arbitrary state ρ^{BR} (for some reference system R), we calculate the result of applying $\mathsf{PauliFilter}^A_S$ to ρ. The state in BR corresponding to $|0\rangle\langle 0|$ in the flag register F is:

$$\begin{aligned}
&\mathrm{Tr}_{AA'}\left[\Pi U^{AB}\left(|\Phi\rangle\langle\Phi|^{AA'} \otimes \rho^{BR}\right)U^\dagger\right] \\
&= \sum_{\substack{(a,b)\in S \\ x,z,x',z'}} \mathrm{Tr}_{AA'}\left[\mathsf{X}^a\mathsf{Z}^b\,|\Phi\rangle\langle\Phi|\,\mathsf{Z}^b\mathsf{X}^a\mathsf{X}^x\mathsf{Z}^z\,|\Phi\rangle\langle\Phi|\,\mathsf{Z}^{z'}\mathsf{X}^{x'}\right] \otimes U^B_{x,z}\rho^{BR}U^\dagger_{x',z'} \\
&= \sum_{\substack{(a,b)\in S \\ x,z,x',z'}} \mathrm{Tr}_{AA'}\left[|\Phi\rangle\langle\Phi|\,\mathsf{X}^{a\oplus x}\mathsf{Z}^{b\oplus z}\,|\Phi\rangle\langle\Phi|\,\mathsf{Z}^{b\oplus z'}\mathsf{X}^{a\oplus x'}\right] \otimes U^B_{x,z}\rho^{BR}U^\dagger_{x',z'}\cdot(-1)^{b\cdot(x\oplus x')} \\
&= \sum_{(a,b)\in S} U^B_{a,b}\rho^{BR}U^\dagger_{a,b}.
\end{aligned}$$

The calculation for the $|1\rangle\langle 1|$-flag is very similar, after observing that

$$\mathbb{I} - \sum_{(a,b)\in S} \mathsf{X}^a\mathsf{Z}^b\,|\Phi\rangle\langle\Phi|\,\mathsf{Z}^b\mathsf{X}^a = \sum_{(a,b)\notin S} \mathsf{X}^a\mathsf{Z}^b\,|\Phi\rangle\langle\Phi|\,\mathsf{Z}^b\mathsf{X}^a.$$

□

C Security of Clifford code

Lemma C.1 (Variation on [AM17, Theorem 3.7]). *Let* M, T, *and* R *be registers with* $\log|M| = 1$ *and* $\log|T| = n > 0$. *Let* U^{MTR} *be a unitary, and write* $U = \sum_{x,z\in\{0,1\}^{n+1}} (\mathsf{X}^x\mathsf{Z}^z)^{MT} \otimes U^R_{x,z}$. *Then for any state* ρ^{MR},

$$\left\| \mathcal{T}^{MT}_{\mathscr{C}_{n+1}}(U)\left(\rho^{MR} \otimes |0^n\rangle\langle 0^n|^T\right) - \left(U^R_{0,0}\rho U^\dagger_{0,0} \otimes |0^n\rangle\langle 0^n|^T + \mathrm{Tr}_M\left[\sum_{(x,z)\neq(0,0)} U^R_{x,z}\rho U^\dagger_{x,z}\right] \otimes \tau^{MT}\right)\right\|_1$$
$$\leqslant \mathsf{negl}(n).$$

Proof. The proof is a straightforward application of the Clifford twirl [ABOEM17, Lemma 3.6], which is similar to Lemma A.2, but for the Clifford group. Using this Clifford twirl in the first

step, and writing $U = \sum_{x,z} (\mathsf{X}^x\mathsf{Z}^z)^{MT} \otimes U^R_{x,z}$, we derive

$$\begin{aligned}
&\mathcal{T}^{MT}_{\mathscr{C}_{n+1}}(U)(\rho \otimes |0^n\rangle\langle 0^n|) \\
&= \sum_{x,z} \mathop{\mathbb{E}}_{C}(C\mathsf{X}^x\mathsf{Z}^z C^\dagger \otimes U_{x,z})(\rho \otimes |0^n\rangle\langle 0^n|)(C^\dagger \mathsf{X}^x\mathsf{Z}^z C \otimes U^\dagger_{x,z}) \\
&= U^R_{0,0}\rho U^\dagger_{0,0} + \sum_{(x,z)\neq(0,0)} \mathop{\mathbb{E}}_{C}(C\mathsf{X}^x\mathsf{Z}^z C^\dagger \otimes U_{x,z})(\rho \otimes |0^n\rangle\langle 0^n|)(C^\dagger \mathsf{X}^x\mathsf{Z}^z C \otimes U^\dagger_{x,z}) \qquad (2) \\
&= U^R_{0,0}\rho U^\dagger_{0,0} + \sum_{(x,z)\neq(0,0)} \mathop{\mathbb{E}}_{(x',z')\neq(0,0)}(\mathsf{X}^{x'}\mathsf{Z}^{z'} \otimes U_{x,z})(\rho \otimes |0^n\rangle\langle 0^n|)(\mathsf{X}^{x'}\mathsf{Z}^{z'} \otimes U^\dagger_{x,z}) \qquad (3) \\
&\approx_{\mathsf{negl}(n)} U^R_{0,0}\rho U^\dagger_{0,0} + \sum_{(x,z)\neq(0,0)} U^R_{x,z}\left(\mathcal{T}^{MT}_{\mathscr{P}_{n+1}}(\rho \otimes |0^n\rangle\langle 0^n|)\right) U^\dagger_{x,z} \\
&= U^R_{0,0}\rho U^\dagger_{0,0} + \sum_{(x,z)\neq(0,0)} \mathrm{Tr}_M\left[U^R_{x,z}\rho U^\dagger_{x,z}\right] \otimes \tau^{MT}.
\end{aligned}$$

In the step from Equation (2) to Equation (3), we used the fact that any non-identity Pauli is mapped to a random non-identity Pauli by expectation over the Clifford group. □

Corollary C.2. *The Clifford authentication code with n trap qubits is* $\mathsf{negl}(n)$*-secure.*

Proof. In the decoding procedure for the n-trap Clifford code, the register T is measured using the two-outcome measurement defined by the projector $\Pi := |0^n\rangle\langle 0^n|$. Note that, given an attack $\mathcal{A}$,

$$\mathop{\mathbb{E}}_{k\in\mathcal{K}}\left[\mathsf{Dec}_k\left(\mathcal{A}\left(\mathsf{Enc}_k\left(\rho\right)\right)\right)\right] = \mathcal{L}^{\Pi}\left(\mathcal{T}_{\mathscr{C}_{n+1}}(\mathcal{A})\left(\rho^{MR} \otimes |0^n\rangle\langle 0^n|^T\right)\right),$$

where $\mathcal{L}^{\Pi}(X) := \mathrm{Tr}_T[\Pi X \Pi] + |\perp\rangle\langle\perp|^M \otimes \mathrm{Tr}_{MT}[\overline{\Pi} X \overline{\Pi}]$. Then apply Lemma C.1, and use the fact that $\mathrm{Tr}[|0\rangle\langle 0|^T \tau^{MT}] = 2^{-n}$. In the terminology of Definition 2.4, we may explicitly describe $\Lambda_{\mathsf{acc}} := U_{0,0}(\cdot)U^\dagger_{0,0}$ and $\Lambda_{\mathsf{rej}} := \sum_{(x,z)\neq(0,0)} U_{x,z}(\cdot)U^\dagger_{x,z}$ for $\mathcal{A} = U(\cdot)U^\dagger$ and U decomposed as in the proof of Lemma C.1. □

D Proof of Lemma 2.7

For the rest of this section, fix a basis $\left|\hat{0}\right\rangle := |\mathsf{T}\rangle$ and $\left|\hat{1}\right\rangle := \left|\mathsf{T}^\perp\right\rangle$. For $w \in \{0,1\}^m$, we will let $|\hat{w}\rangle := |\hat{w}_1\rangle \ldots |\hat{w}_m\rangle$. Then the all-0s string in this basis represents m copies of $|\mathsf{T}\rangle$. We analyze Circuit 2.8. It is simple to verify that $\hat{\mathsf{Z}} = \left|\hat{0}\right\rangle\left\langle\hat{0}\right| - \left|\hat{1}\right\rangle\left\langle\hat{1}\right|$ (up to a global phase). The first step of the circuit is to apply $\hat{\mathsf{Z}}$ with probability $\frac{1}{2}$ to each qubit, which has the effect of *dephasing* the qubit, or equivalently, making the state diagonal, in the $\{\left|\hat{0}\right\rangle, \left|\hat{1}\right\rangle\}$ basis. More precisely, if we let $\rho = \sum_{w,w'\in\{0,1\}^m} \alpha_{w,w'} |\hat{w}\rangle\langle\hat{w}'|$:

$$\begin{aligned}
\rho &\mapsto \sum_{w,w'\in\{0,1\}^m} \alpha_{w,w'} \bigotimes_{i=1}^m \frac{1}{2}\left(|\hat{w}_i\rangle\left\langle\hat{w}'_i\right| + \hat{\mathsf{Z}}\,|\hat{w}_i\rangle\left\langle\hat{w}'_i\right|\hat{\mathsf{Z}}\right) \qquad (4) \\
&= \sum_{w,w'\in\{0,1\}^m} \alpha_{w,w'} \bigotimes_{i=1}^m \frac{1}{2}\left(|\hat{w}_i\rangle\left\langle\hat{w}'_i\right| + (-1)^{w_i+w'_i}|\hat{w}_i\rangle\left\langle\hat{w}'_i\right|\right) \qquad (5) \\
&= \sum_{w\in\{0,1\}^m} \alpha_{w,w}\,|\hat{w}\rangle\langle\hat{w}| =: \rho'. \qquad (6)
\end{aligned}$$

Let Ξ' denote the quantum channel given by steps 2–3 of Circuit 2.8. Note that Ξ' is symmetric: Given two inputs ρ and $\rho' = \pi\rho\pi^\dagger$, after step 2, either state will be mapped to $\sum_{\tau\in S_m} \frac{1}{m!}\tau\rho\tau^\dagger$. Thus, we can apply Theorem D.1 of [DNS10], which states the following:

Theorem D.1 ([DNS10]). *Let σ be an m-qubit state, diagonal in the basis $\{|\hat{w}\rangle : w \in \{0,1\}^m\}$, and suppose $\Pi_{LW}\sigma = \sigma$. Let Ξ' be any CPTP map from m qubits to t qubits such that $\Xi'(\pi\omega\pi^\dagger) = \Xi'(\omega)$ for any n-qubit state ω and any $\pi \in S_m$. Then, letting $\delta_s = \frac{s}{m}$:*

$$\| \Xi'(\sigma) - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \|_1 \leqslant (m+1) \max_{s\leqslant\ell} \left\| \Xi'\left(((1-\delta_s)|\hat{0}\rangle\langle\hat{0}| + \delta_s |\hat{1}\rangle\langle\hat{1}|)^{\otimes m} \right) - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \right\|_1 .$$

We can put everything together to prove Lemma 2.7.

of Lemma 2.7. Let ρ' be as in (6). We have

$$\mathrm{Tr}(\Pi_{LW}\rho') = \mathrm{Tr}(\Pi_{LW}\rho) \geqslant 1-\varepsilon.$$

Thus, write $\rho' = (1-\varepsilon)\sigma + \varepsilon\sigma'$ for $\sigma = \frac{1}{1-\varepsilon}\Pi_{LW}\rho'$. Applying Theorem D.1, we get

$$\left\| \Xi'(\sigma) - \left(|\hat{0}\rangle\langle\hat{0}|\right)^{\otimes t} \right\|_1 \leqslant (m+1) \max_{s\leqslant\ell} \left\| \Xi'\left(((1-\delta_s)|\hat{0}\rangle\langle\hat{0}| + \delta_s |\hat{1}\rangle\langle\hat{1}|)^{\otimes m} \right) - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \right\|_1 .$$

On a symmetric state, Ξ' is simply the state distillation protocol of [BK05], applied t times in parallel to m/t qubits each time. Let Φ be one state distillation protocol distilling one qubit from m/t (so Ξ' acts as $\Phi^{\otimes t}$ on symmetric states). By Theorem 2.6, using $\delta_s = \frac{s}{m} \leqslant \frac{\ell}{m}$, and

$$\tau = \left((1-\delta_s)|\hat{0}\rangle\langle\hat{0}| + \delta_s|\hat{1}\rangle\langle\hat{1}|\right)^{\otimes m/t},$$

we have

$$1 - \langle\hat{0}|\Phi(\tau)|\hat{0}\rangle \leqslant O\left((5\delta_s)^{(m/t)^c}\right) \leqslant O\left(\left(5\frac{\ell}{m}\right)^{(m/t)^c}\right)$$

for $c \approx .2$. Let $\delta = (5\ell/m)^{(m/t)^c}$. Using the inequality between trace distance and fidelity, $\|\rho - |\psi\rangle\langle\psi|\|_1 \leqslant 2\sqrt{1-\langle\psi|\rho|\psi\rangle}$, we have

$$\begin{aligned}
&\left\| \Xi'\left(((1-\delta_s)|\hat{0}\rangle\langle\hat{0}| + \delta_s|\hat{1}\rangle\langle\hat{1}|)^{\otimes m} \right) - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \right\|_1 \\
&= \| \Phi(\tau)^{\otimes t} - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \|_1 \\
&\leqslant 2\sqrt{1 - (\langle\hat{0}|\Phi(\tau)|\hat{0}\rangle)^t} \leqslant 2\sqrt{1-(1-\delta)^t}.
\end{aligned}$$

Since $1-x \leqslant e^{-x}$ for all x, and $e^{-2x} \leqslant 1-x$ whenever $x \leqslant 1/2$, it follows that:

$$1 - 2\delta t \leqslant e^{-2\delta t} \leqslant (1-\delta)^t.$$

Thus

$$\left\| \Xi'\left(((1-\delta_s)|\hat{0}\rangle\langle\hat{0}| + \delta_s|\hat{1}\rangle\langle\hat{1}|)^{\otimes m} \right) - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \right\|_1 \leqslant 2\sqrt{2\delta t}.$$

Thus, we have:

$$\begin{aligned}
\| \Xi(\rho) - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \|_1 = \| \Xi'(\rho') - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \|_1 &= \| (1-\varepsilon)\Xi'(\sigma) + \varepsilon\Xi'(\sigma') - (|\hat{0}\rangle\langle\hat{0}|)^{\otimes t} \|_1 \\
&\leqslant (1-\varepsilon)(m+1)2\sqrt{2\delta t} + \varepsilon = O\left(m\sqrt{t}(5\ell/m)^{(m/t)^c/2} + \varepsilon\right).
\end{aligned}$$

□

E Proof of Lemma 4.3

Before we prove Lemma 4.3, let us begin by zooming in on the test phase (steps 4–6): we show in a separate lemma that, with high probability, it only checks out if the resulting state is a correctly-encoded one.

For a projector Π on two n-qubit quantum registers T_1T_2, define the quantum channel $\mathcal{L}^{\Pi}$ by

$$\mathcal{L}^{\Pi}(X) = \Pi(X)\Pi + |\perp\rangle\langle\perp| \operatorname{Tr}\left[\bar{\Pi}X\right]$$

where $|\perp\rangle$ is a distinguished state on T_1T_2 with $\Pi|\perp\rangle = 0$. Furthermore, for $x \in \{0,1\}^n$, define the "full" and "half" projectors

$$\Pi_{s,F} := \begin{cases} \left|0^{2n}\right\rangle\left\langle 0^{2n}\right| & s = 0 \\ 0 & \text{else} \end{cases}$$
$$\Pi_{s,H} := \mathbb{I} \otimes |s\rangle\langle s| .$$

The following lemma shows that measuring $\Pi_{s,F}$ on the one hand, and applying a twirl $\mathcal{T}_{\mathrm{GL}(2n,\mathbb{F}_2)}$ followed by a measurement of $\Pi_{s,H}$ are equivalent as tests in the above protocol.

Lemma E.1. *Applying a random element of* $\mathrm{GL}(2n,\mathbb{F}_2)$ *followed by* $\mathcal{L}^{\Pi_{s,H}}$ *is essentially equivalent to applying* $\mathcal{L}^{\Pi_{s,F}}$*:*

$$\left\|\mathcal{L}^{\Pi_{s,F}} - \mathcal{L}^{\Pi_{s,H}} \circ \mathcal{T}_{\mathrm{GL}(2n,\mathbb{F}_2)}\right\|_{\diamond} \leqslant 12 \cdot 2^{-\frac{n}{2}} \leqslant \mathsf{negl}\,(n) .$$

Proof. First, observe the following facts about a random $g \in \mathrm{GL}(2n,\mathbb{F}_2)$. Of course, $g0 = 0$ by linearity. On the other hand, gx is uniformly random on $\mathbb{F}_2^{2n} \setminus \{0\}$ for $x \neq 0$. More generally, x and y with $x \neq 0 \neq y$ are linearly independent if and only if $x \neq y$, and therefore (gx, gy) is uniformly random on $\left\{(x,y) \in \left(\mathbb{F}_2^{2n} \setminus \{0\}\right)^2 | x \neq y\right\}$. In the following we abbreviate $\mathcal{T} := \mathcal{T}_{\mathrm{GL}(2n,\mathbb{F}_2)}$. We calculate for $x, y \in \mathbb{F}_2^{2n} \setminus \{0\}$ with $x \neq y$,

$$\begin{aligned}
\mathcal{T}(|0\rangle\langle 0|) &= |0\rangle\langle 0| \\
\mathcal{T}(|x\rangle\langle x|) &= \frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1} \\
\mathcal{T}(|x\rangle\langle 0|) &= \left(2^{2n}-1\right)^{-\frac{1}{2}} \left|+'\right\rangle\langle 0| \\
\mathcal{T}(|x\rangle\langle y|) &= \left(2^{2n}-1\right)^{-1}\left(2^{2n}-2\right)^{-1} \sum_{\substack{z,t \in \mathbb{F}_2^{2n}\setminus\{0\} \\ z \neq t}} |z\rangle\langle t| \\
&= \left(2^{2n}-2\right)^{-1}\left(\left|+'\right\rangle\left\langle+'\right| - \frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1}\right) =: S.
\end{aligned}$$

Here we have defined the unit vector

$$\left|+'\right\rangle = \left(2^{2n}-1\right)^{-\frac{1}{2}} \sum_{x \in \mathbb{F}_2^{2n}\setminus\{0\}} |x\rangle .$$

We can now evaluate T on an arbitrary input density matrix,

$$\begin{aligned}\mathcal{T}(\rho_{T_1T_2E}) = &\sum_{x,y\in\{0,1\}^{2n}} \mathcal{T}(|x\rangle\langle y|)_{T_1T_2} \otimes \langle x|_{T_1T_2}\, \rho_{T_1T_2E}\, |y\rangle_{T_1T_2} \\ = &|0\rangle\langle 0|_{T_1T_2} \otimes \langle 0|_{T_1T_2}\, \rho_{T_1T_2E}\, |0\rangle_{T_1T_2} + |0\rangle\langle +'|_{T_1T_2} \otimes \langle 0|_{T_1T_2}\, \rho_{T_1T_2E}\, |+'\rangle_{T_1T_2} \\ &+ |+'\rangle\langle 0|_{T_1T_2} \otimes \langle +'|_{T_1T_2}\, \rho_{T_1T_2E}\, |0\rangle_{T_1T_2} + \frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1} \otimes \mathrm{Tr}_{T_1T_2}[(\mathbb{I} - |0\rangle\langle 0|)_{T_1T_2}\rho_{T_1T_2E}] \\ &+ S \otimes \sum_{\substack{x,y\in\mathbb{F}_2^{2n}\setminus\{0\} \\ x\neq y}} \langle x|_{T_1T_2}\, \rho_{T_1T_2E}\, |y\rangle_{T_1T_2}\,. \end{aligned} \tag{7}$$

We calculate

$$\|\Pi_{s,H}\,|+'\rangle\,\|_2 \leqslant \sqrt{\frac{2^n}{2^{2n}-1}} \quad \text{and} \quad \left\|\Pi_{s,H}\frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1}\Pi_{s,H}\right\|_1 \leqslant \frac{2^n-1}{2^{2n}-1},$$

where the first inequality is tight for $s = 0$ and the second inequality for $s \neq 0$. We also have

$$\|S\|_1 = 2\frac{1 - \left(2^{2n}-1\right)^{-1}}{2^{2n}-2}$$

Using these quantities, we analyze $\Pi_{s,H}\mathcal{T}(\rho_{T_1T_2E})\Pi_{s,H}$ term by term according to equation (7). We have

$$\Pi_{s,H}\,|0\rangle\langle 0|_{T_1T_2}\,\Pi_{s,H} = \begin{cases} |0\rangle\langle 0|_{T_1T_2} & s = 0 \\ 0 & \text{else,} \end{cases} \tag{8}$$

$$\left\|\Pi_{s,H}\,|0\rangle\langle +'|_{T_1T_2}\,\Pi_{s,H} \otimes \langle 0|_{T_1T_2}\,\rho_{T_1T_2E}\,|+'\rangle_{T_1T_2}\right\|_1 \leq \sqrt{\frac{2^n}{2^{2n}-1}}, \tag{9}$$

$$\left\|\Pi_{s,H}\frac{\mathbb{I} - |0\rangle\langle 0|}{2^{2n}-1}\Pi_{s,H} \otimes \mathrm{Tr}_{T_1T_2}[(\mathbb{I} - |0\rangle\langle 0|)_{T_1T_2}\rho_{T_1T_2E}]\right\|_1 \leq \frac{2^n}{2^{2n}-1}\text{ , and} \tag{10}$$

$$\tag{11}$$

$$\left\|\Pi_{s,H}S\Pi_{s,H}\otimes\sum_{\substack{x,y\in\mathbb{F}_2^{2n}\setminus\{0\}\\x\neq y}}\langle x|_{T_1T_2}\,\rho_{T_1T_2E}\,|y\rangle_{T_1T_2}\right\|_1\leq\left\|\Pi_{s,H}S\Pi_{s,H}\otimes\sum_{x,y\in\mathbb{F}_2^{2n}\setminus\{0\}}\langle x|_{T_1T_2}\,\rho_{T_1T_2E}\,|y\rangle_{T_1T_2}\right\|_1$$
$$+\left\|\Pi_{s,H}S\Pi_{s,H}\otimes\sum_{x\in\mathbb{F}_2^{2n}\setminus\{0\}}\langle x|_{T_1T_2}\,\rho_{T_1T_2E}\,|x\rangle_{T_1T_2}\right\|_1$$
$$\leq 2\left(2^{2n}-2\right)^{-1}\left(\left\|\Pi_{s,H}\left|+'\right\rangle\!\left\langle+'\right|\Pi_{s,H}\right\|_1+\left\|\Pi_{s,H}\frac{\mathbb{I}-|0\rangle\langle 0|}{2^{2n}-1}\Pi_{s,H}\right\|_1\right)$$
$$\mathrm{Tr}\left[\left(\left(2^{2n}-1\right)\left|+'\right\rangle\!\left\langle+'\right|+\mathbb{I}-|0\rangle\langle 0|\right)_{T_1T_2}\rho_{T_1T_2E}\right]$$
$$\leq 2\left(2^{2n}-2\right)^{-1}\left(\frac{2^n}{2^{2n}-1}+\frac{2^n}{2^{2n}-1}\right)\left(2\left(2^{2n}-1\right)\right]$$
$$\leq 8\frac{2^n\left(2^{2n}-1\right)}{\left(2^{2n}-1\right)\left(2^{2n}-2\right)}$$
$$\leq 2\frac{2^n}{\left(2^{2n}-2\right)}. \tag{12}$$

The first, second and fifth inequality use normalization of ρ and the third and fourth inequality use the triangle inequality for the trace norm. The fifth inequality additionally uses Hölder's inequality. The observation that

$$\Pi_{s,F}\mathcal{T}(\rho_{T_1T_2E})\Pi_{s,F}=\begin{cases}|0\rangle\langle 0|_{T_1T_2}\otimes\langle 0|_{T_1T_2}\,\rho_{T_1T_2E}\,|0\rangle_{T_1T_2} & s=0\\ 0 & \text{else}\end{cases}, \tag{13}$$

together with Equations (7) and (8)-(12), we get that there exists a $\rho_{T_1T_2E}$ such that

$$\begin{aligned}\left\|\mathcal{L}^{\Pi_{s,F}}-\mathcal{L}^{\Pi_{s,H}}\circ\mathcal{T}_{\mathrm{GL}(2n,\mathbb{F}_2)}\right\|_\diamond=&\left\|\mathcal{L}^{\Pi_{s,F}}(\rho)-\mathcal{L}^{\Pi_{s,H}}\circ\mathcal{T}_{\mathrm{GL}(2n,\mathbb{F}_2)}(\rho)\right\|_1\\ \leq&\left\|\Pi_{s,H}|0\rangle\left\langle+'\right|_{T_1T_2}\Pi_{s,H}\otimes\langle 0|_{T_1T_2}\,\rho_{T_1T_2E}\left|+'\right\rangle_{T_1T_2}\right\|_1\\ &+\left\|\Pi_{s,H}\left|+'\right\rangle\langle 0|_{T_1T_2}\Pi_{s,H}\otimes\left\langle+'\right|_{T_1T_2}\rho_{T_1T_2E}\,|0\rangle_{T_1T_2}\right\|\\ &+\left\|\Pi_{s,H}\frac{\mathbb{I}-|0\rangle\langle 0|}{2^{2n}-1}\Pi_{s,H}\otimes\mathrm{Tr}_{T_1T_2}[(\mathbb{I}-|0\rangle\langle 0|)_{T_1T_2}\rho_{T_1T_2E}]\right\|_1\\ &+\left\|\Pi_{s,H}S\Pi_{s,H}\otimes\sum_{\substack{x,y\in\mathbb{F}_2^{2n}\setminus\{0\}\\x\neq y}}\langle x|_{T_1T_2}\,\rho_{T_1T_2E}\,|y\rangle_{T_1T_2}\right\|_1\\ \leq&2\sqrt{\frac{2^n}{2^{2n}-1}}+\frac{2^n}{2^{2n}-1}+8\frac{2^n}{2^{2n}-2}\\ \leq&12\cdot 2^{-\frac{n}{2}},\end{aligned}$$

which is negligible. □

Now that we have established that it suffices to measure only the T_2 register (after applying a random $g\in GL(2n,\mathbb{F}_2)$), we are ready to prove the security of Protocol 4.2:

of Lemma 4.3. We consider two cases: either player 1 is honest, or she is corrupted.

Case 1: player 1 is honest. Assume, without loss of generality, that all other players are corrupted: $I_{\mathcal{A}} = \{2, 3, \ldots, k\}$. That this is indeed the worst case can be seen as follows. The situation where a set H of additional players are honest can be emulated by an adversary in the case where only player 1 is honest, but follows the protocol for the additional honest players, and does not use any information received by those honest players as part of the dishonest players' (i.e., the players in $I_{\mathcal{A}} \setminus H$) actions.

The corrupted players act as one entity whose honest action is to apply $U_H := F_k F_{k-1} \cdots F_3 F_2$, and return the state to player 1. Without loss of generality, assume that $\mathcal{A}$ is unitary by expanding the side-information register R as necessary. Then, define an attack unitary $A := U_H^\dagger \mathcal{A}$, so that we may write $\mathcal{A} = U_H A$. In other words, we establish that $\mathcal{A}$ consists of a unitary attack A, followed by the honest unitary U_H. Note that A may depend arbitrarily on its instructions F_2 through F_k

The simulator $\mathcal{S}$ has access to the ideal functionality only through the ability to submit the bits b_i for players $i \neq 1$. It does not receive any input from the environment, except for a side-information register R. Define the simulator as follows (in terms of an adversarial map A):

Simulator 1. On input register R received from the environment, do:

1. Sample random $F'_1, F'_2, \ldots, F'_k \in \mathscr{C}_{2n+1}$.[a]
2. Run $\mathsf{IdFilter}^{MT_1}(A)$ on the register R, using the instructions $F'_2, F'_3, \ldots, F'_k$ to determine A. (See Section 2.3.)
3. If the flag register is 0, set $b_i = 0$ for all $i \neq 1$. Otherwise, set $b_i = 1$. Submit the bits b_i to the ideal functionality.

[a] Whenever a simulator samples random elements, it does so by running the ideal functionality for classical MPC with the adversary it is currently simulating. If that ideal functionality aborts, the simulator will also abort by setting $b_i = 1$ for the adversarial players i. In that case, the simulated output state and the real output state will be indistinguishable by security of the classical MPC. To avoid clutter in the exposition of our simulators and proofs, we will ignore this technicality, and pretend that the simulator generates the randomness itself.

We will consider the joint state in the output register $R_1^{\mathsf{out}} = MT_1$, the state register S, and the attacker's side-information register R in both the real and the ideal (simulated) case. In both cases, it will be useful to decompose the attack map A as

$$A = \sum_{a,c \in \{0,1\}^{n+1}} (\mathsf{X}^a \mathsf{Z}^c)^{MT_1} \otimes A^R_{a,c}.$$

We start by analyzing the ideal case. By Lemma 2.3, and using $\mathcal{P} = \{(0,0)\}$ with 0 as an abbreviation for 0^n, the output state in MT_1RS in case of accept (setting all $b_i = 0$) is

$$\mathop{\mathbb{E}}_E E^{MT_1} \left(A^R_{0,0} \rho^{MR} A^\dagger_{0,0} \otimes |0^n\rangle\langle 0^n|^{T_1} \right) E^\dagger \otimes |E\rangle\langle E|^S . \tag{14}$$

The output state in MT_1RS in case of reject is (again by Lemma 2.3)

$$\mathcal{T}^{MT_1}_{\mathscr{C}_{n+1}}\left(\sum_{(x,z)\neq(0,0)} A^R_{x,z}\rho^{MR}A^\dagger_{x,z}\otimes|0^n\rangle\langle 0^n|^{T_1}\right)\otimes|\perp\rangle\langle\perp|^S \tag{15}$$

$$=\tau^{MT_1}\otimes\sum_{(x,z)\neq(0,0)} A_{x,z}\rho_R A^\dagger_{x,z}\otimes|\perp\rangle\langle\perp|^S\,. \tag{16}$$

Next, we consider the state in MT_1RS after the real protocol is executed, and argue that it is negligibly close to Equations (14)+(16). Again, we first consider the accept case. Following the steps in Protocol 4.2 on an input state ρ^{MR}, and noting that

$$\left(F_1^\dagger\right)^{MT_1T_2}A^{MT_1T_2R}F_1\left(\rho^{MR}\otimes\left|0^{2n}\right\rangle\left\langle 0^{2n}\right|^{T_1T_2}\right)F_1^\dagger A^\dagger F_1=\left(\mathcal{T}^{MT_1T_2}_{\mathscr{C}_{2n+1}}(A)\right)\left(\rho\otimes\left|0^{2n}\right\rangle\left\langle 0^{2n}\right|\right),$$

the output state in the case of accept is

$$\begin{aligned}
&\mathop{\mathbb{E}}_{E,r,s}\langle r|^{T_2}(E\otimes\mathsf{X}^r\mathsf{Z}^s)\,\mathcal{T}^{T_1T_2}_{\mathrm{GL}_{(2n,\mathbb{F}_2)}}\left(\left(\mathcal{T}^{MT_1T_2}_{\mathscr{C}_{2n+1}}(A)\right)\left(\rho\otimes\left|0^{2n}\right\rangle\left\langle 0^{2n}\right|\right)\right)(E\otimes\mathsf{X}^r\mathsf{Z}^s)^\dagger|r\rangle\otimes|E\rangle\langle E|^S\\
&=\mathop{\mathbb{E}}_{E}E^{MT_1}\langle 0^n|^{T_2}\,\mathcal{T}^{T_1T_2}_{\mathrm{GL}_{(2n,\mathbb{F}_2)}}\left(\left(\mathcal{T}^{MT_1T_2}_{\mathscr{C}_{2n+1}}(A)\right)\left(\rho\otimes\left|0^{2n}\right\rangle\left\langle 0^{2n}\right|\right)\right)|0^n\rangle\,E^\dagger\otimes|E\rangle\langle E|^S\\
&\approx_{\mathsf{negl}(n)}\mathop{\mathbb{E}}_{E}E^{MT_1}\left\langle 0^{2n}\right|^{T_1T_2}\left(\left(\mathcal{T}^{MT_1T_2}_{\mathscr{C}_{2n+1}}(A)\right)\left(\rho\otimes\left|0^{2n}\right\rangle\left\langle 0^{2n}\right|\right)\right)\left|0^{2n}\right\rangle E^\dagger\otimes|E\rangle\langle E|^S,
\end{aligned} \tag{17}$$

where the approximation follows from Lemma E.1. This is where the authentication property of the Clifford code comes in: by Lemma C.1, only the part of A that acts trivially on MT_1T_2 remains after the measurement of T_1T_2. Thus, Eq. (17) $\approx_{\mathsf{negl}(n)}$ Eq. (14).

The reject case of the real protocol is similar: again using Lemmas E.1 and C.1, we can see that it approximates (up to a negligible factor in n) Eq. (16).

We conclude that, from the point of view of any environment, the real output state in registers MT_1SR (encoding, memory state, and side information) is indistinguishable from the simulated state.

Case 2: player 1 is dishonest. Assume (without loss of generality) that the only honest player is player 2, i.e., $I_A=\{1,3,4,...,k\}$.

In the real protocol, the adversary interacts with the honest player 2, and has two opportunities to attack: before player 2 applies its Clifford operation, and after.

The adversaries' actions before the interaction with player 2 can, without loss of generality, be described by a unitary $U_{H,A}A$, that acts on the input state ρ^{MR}, plus the registers T_1T_2 that are are initialized to zero. The unitary $U_{H,A}$ is player 1's honest operation $F_1^{MT_1T_2}$.

Similarly, the adversaries' actions after the interaction with player 2 can be described by a unitary $BU_{H,B}$, followed by a computational-basis measurement on T_2 which results in an n-bit string r'. Again, $U_{H,B}$ is the honest unitary $VF_k\cdots F_4F_3$ that should be applied jointly by players $3,4,\ldots,k,1$.

For any adversary, described by such unitaries A and B, define a simulator as follows:

Simulator 2. On input register MR received from the environment, do:

1. Initialize $b_i = 0$ for all $i \in I_A$.
2. Sample random $F_1, F_2, \ldots, F_k \in \mathscr{C}_{2n+1}$. Run $\mathsf{ZeroFilter}^{T_1T_2}(A)$ on MR, using the instructions F_i (for all $i \in I_{\mathcal{A}}$) to determine A. If the filter flag is 1, abort by setting $b_1 = 1$.
3. Input the M register into the ideal functionality, and receive a state in the register MT_1.
4. Run $\mathsf{XFilter}^{T_2}(B)$ on MT_1R, using the instruction $V := F_2^\dagger F_3^\dagger \cdots F_k^\dagger$ to determine B. (This choice of V ensures that the honest action $U_{H,B}$ is identity.) If the filter flag is 1, abort by setting $b_1 = 1$.
5. Submit all the b_i to the ideal functionality.

Similarly to the previous case, we consider the output state in the registers MT_1RS in both the ideal (simulated) case, and the real case, as computed on an input state ρ^{MR}.

Again, we decompose the attack maps A and B as

$$A = \sum_{a,c\in\{0,1\}^{2n}} (\mathsf{X}^a\mathsf{Z}^c)^{T_1T_2} \otimes A_{a,c}^{MR} \quad , \tag{18}$$

$$B = \sum_{b,d\in\{0,1\}^{n}} \left(\mathsf{X}^b\mathsf{Z}^d\right)^{T_2} \otimes B_{b,d}^{MT_1R} \quad . \tag{19}$$

Note that the decompositions are taken over different registers for A and B. In the derivations below, we will often abbreviate $A_a := \sum_b A_{a,b}$, and, in the subscripts, we will abbreviate 0 for the all-zero string.

In the ideal or simulated case, one of three things may happen: the simulator sets b_1 to 0 (signaling accept to the ideal functionality), or sets b_1 to 1 in step 2, or sets b_1 to 1 in step 4 (both signaling reject to the ideal functionality). The ideal output state is thus the sum of three separate terms, which we will analyze separately.

We start with the the accept case, where both filters result in a $|0\rangle\langle 0|$ flag. Using the decompositions from Equations (18) and (19), we apply Lemma 2.3 to see that the resulting state is

$$\mathop{\mathbb{E}}_{E} \sum_d B_{0,d}^{MT_1R} E\left(A_0\rho A_0^\dagger \otimes |0^n\rangle\langle 0^n|^{T_1}\right) E^\dagger B_{0,d}^\dagger \otimes |E\rangle\langle E|^S \,. \tag{20}$$

Here, E is the key that the ideal functionality samples (and stores in the register S) when it is called to encode M.

Next, we consider the simulator choosing $b_1 = 1$ already in step 2, the zero filter has failed. In this case, the ideal functionality does not store the encoding key E in the register S. This allows us to view the Clifford encoding as a twirl on the Clifford group. The output state is (by Lemma 2.3)

$$\begin{aligned} &\sum_{a\neq 0^{2n},b,d} B_{b,d}\mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1}\left(A_a\rho^{MR}A_a^\dagger \otimes |0^n\rangle\langle 0^n|^{T_1}\right) B_{b,d}^\dagger \otimes |\perp\rangle\langle\perp|^S \\ = &\sum_{a\neq 0^{2n},b,d} B_{b,d}\left(\mathrm{Tr}_M\left[A_a\rho^{MR}A_a^\dagger\right] \otimes \tau^{MT_1}\right) B_{b,d}^\dagger \otimes |\perp\rangle\langle\perp|^S \,. \end{aligned} \tag{21}$$

Note that in this case, the flag in the X filter does not influence the bit b_1 (it is already set to 1). Therefore, both terms in Lemma 2.3 survive, and all pairs (b, d) are included in the sum.

Finally, we look at the case where the zero filter does not result in changing b_1, but the X filter does, in step 4. If this happens, the key E is erased so we can again apply a Clifford twirl, and the output state is (by Lemma 2.3)

$$\begin{aligned}&\sum_{b\neq 0^n,d} B_{b,d}\mathcal{T}^{MT_1}_{\mathscr{C}_{n+1}}\left(A_0\rho^{MR}A_0^\dagger\otimes|0^n\rangle\langle 0^n|^{T_1}\right)B_{b,d}^\dagger\otimes|\perp\rangle\langle\perp|^S\\ =&\sum_{b\neq 0^n} B_{b,d}\left(\mathrm{Tr}_M\left[A_0\rho^{MR}A_0^\dagger\right]\otimes\tau^{MT_1}\right)B_{b,d}^\dagger\otimes|\perp\rangle\langle\perp|^S.\end{aligned}\tag{22}$$

In summary, the output state in the ideal case is

$$\text{Eq. (20)} + \text{Eq. (21)} + \text{Eq. (22)}.$$

In the real protocol, only one measurement is performed at the end. The output state in the real case is thus a sum of only two terms: an accept and reject case. We will again analyze these separately, and will show that the accept state is approximately equal to Equation (20), while the reject state approximates Equations (21) + (22).

Following Protocol 4.2 on an input state ρ^{MR}, and canceling out the F_i and $F_i^\dagger$ terms that are part of the honest actions, we first consider the state in case of accept. We abbreviate

$$\begin{aligned}\sigma :=&\mathop{\mathbb{E}}_g E^{MT_1}U_g^{T_1T_2}(A(\rho\otimes\left|0^{2n}\right\rangle\left\langle 0^{2n}\right|)A^\dagger)U_g^\dagger E^\dagger\\ =&\quad E^{MT_1}\mathcal{T}^{T_1T_2}_{GL(2n,\mathbb{F}_2)}(A(\rho\otimes\left|0^{2n}\right\rangle\left\langle 0^{2n}\right|)A^\dagger)E^\dagger,\end{aligned}$$

where we are allowed to view $\mathbb{E}_g U_g(\cdot)U_g^\dagger$ as a Twirling operation, since A and B are independent of g. We decompose the attack B as in Equation (19), and derive the accept case

$$\mathop{\mathbb{E}}_{E,r,s}\langle r|^{T_2}B\left(\mathsf{X}^r\mathsf{Z}^s\right)^{T_2}\sigma\left(\mathsf{X}^r\mathsf{Z}^s\right)^\dagger B^\dagger|r\rangle\otimes|E\rangle\langle E|^S\tag{23}$$

$$=\mathop{\mathbb{E}}_{E,r,s}\langle 0|^{T_2}\left(\mathsf{X}^r\mathsf{Z}^s\right)^{\dagger T_2}B\left(\mathsf{X}^r\mathsf{Z}^s\right)^{T_2}\sigma\left(\mathsf{X}^r\mathsf{Z}^s\right)^\dagger B^\dagger\left(\mathsf{X}^r\mathsf{Z}^s\right)|0\rangle\otimes|E\rangle\langle E|^S$$

$$=\mathop{\mathbb{E}}_{E,r,s}\sum_{b,d,b',d'}\langle 0|^{T_2}\left(\left(\mathsf{X}^r\mathsf{Z}^s\mathsf{X}^b\mathsf{Z}^d\mathsf{X}^r\mathsf{Z}^s\right)\otimes B_{b,d}\right)\sigma\left(\left(\mathsf{X}^r\mathsf{Z}^s\mathsf{X}^{b'}\mathsf{Z}^{d'}\mathsf{X}^r\mathsf{Z}^s\right)\otimes B_{b',d'}^\dagger\right)|0\rangle\otimes|E\rangle\langle E|^S\tag{24}$$

$$=\mathop{\mathbb{E}}_{E}\sum_{b,d}\langle b|^{T_2}B_{b,d}\sigma B_{b,d}^\dagger|b\rangle\otimes|E\rangle\langle E|^S\tag{25}$$

$$=\mathop{\mathbb{E}}_{E}\sum_{b,d}B_{b,d}\mathrm{Tr}_{T_2}\left[\Pi_{b,H}^{T_2}\sigma\Pi_{b,H}^\dagger\right]B_{b,d}^\dagger\otimes|E\rangle\langle E|^S.\tag{26}$$

From Equation (24) to (25), we used the Pauli twirl to remove all terms for which $(b, d) \neq (b', d')$. This application of the Pauli twirl is possible, because neither A nor B depends on r, s.

We continue with the accept case by expanding σ in Equation (26), and evaluate the effect of the random $\mathrm{GL}_{2n,\mathbb{F}_2}$ element on T_1T_2 using Lemma E.1. It ensures that, if A altered the T_1T_2

register, then B cannot successfully reset the register T_2 to the correct value r. It follows that

$$\text{Eq. (26)} \approx \mathop{\mathbb{E}}_{E} \sum_{b,d} B_{b,d}^{MT_1R} E^{MT_1} \mathrm{Tr}_{T_2} \left[\Pi_{b,F}^{T_1T_2} A \left(\rho \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right| \right) A^\dagger \Pi_{b,F}^\dagger \right] E^\dagger B_{b,d}^\dagger \otimes |E\rangle\langle E|^S \tag{27}$$

$$= \mathop{\mathbb{E}}_{E} \sum_{d} B_{0,d}^{MT_1R} E^{MT_1} \mathrm{Tr}_{T_2} \left[\left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right| A \left(\rho \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right| \right) A^\dagger \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right| \right] E^\dagger B_{0,d}^\dagger \otimes |E\rangle\langle E|^S$$

$$= \mathop{\mathbb{E}}_{E} \sum_{d} B_{0,d}^{MT_1R} E^{MT_1} \left(A_0 \rho A_0^\dagger \otimes |0^n\rangle\langle 0^n| \right) E^\dagger B_{0,d}^\dagger \tag{28}$$

$$= \text{Eq. (20)}. \tag{29}$$

The difference in the approximation is bound by $\mathsf{negl}\,(n)$, since for each b we can use Lemma E.1 (and there is an implicit average over the bs because of the normalization factor induced by $B_{b,d}$ operator). Essentially, the only way to pass the measurement test successfully is for A not to alter the all-zero state in T_1T_2, and for B to leave T_2 unaltered in the computational basis. This is reflected in the simulator's zero filter and X filter, respectively.

If the real protocol rejects, the MPC stores a dummy $\perp$ in the key register S. The resulting state can be derived in a similar way, up to Equation (27), after which the derivation becomes slightly different. The output state in the case of reject approximates (up to a difference of $\mathsf{negl}\,(n)$)

$$\begin{aligned}
&\mathop{\mathbb{E}}_{E} \sum_{b,d} B_{b,d}^{MT_1R} E^{MT_1} \mathrm{Tr}_{T_2} \left[\left(\mathbb{I} - \Pi_{b,F}\right)^{T_1T_2} A \left(\rho \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right| \right) A^\dagger \left(\mathbb{I} - \Pi_{b,F}\right)^\dagger \right] E^\dagger B_{b,d}^\dagger \otimes |\perp\rangle\langle\perp|^S \\
= &\sum_{b,d} B_{b,d}^{MT_1R} \mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1} \left(\mathrm{Tr}_{T_2} \left[\left(\mathbb{I} - \Pi_{b,F}\right)^{T_1T_2} A \left(\rho \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right| \right) A^\dagger \left(\mathbb{I} - \Pi_{b,F}\right)^\dagger \right] \right) B_{b,d}^\dagger \otimes |\perp\rangle\langle\perp|^S
\end{aligned} \tag{30}$$

$$\begin{aligned}
= &\sum_{\substack{b\neq 0^n \\ d,a,a'}} B_{b,d}^{MT_1R} \mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1} \left(\mathrm{Tr}_{T_2} \left[A_a \rho^{MR} A_{a'}^\dagger \otimes |a\rangle\langle a'| \right] \right) B_{b,d}^\dagger \otimes |\perp\rangle\langle\perp|^S \\
&+ \sum_{\substack{a,a'\neq 0^{2n} \\ d}} B_{0,d}^{MT_1R} \mathcal{T}_{\mathscr{C}_{n+1}}^{MT_1} \left(\mathrm{Tr}_{T_2} \left[A_a \rho^{MR} A_{a'}^\dagger \otimes |a\rangle\langle a'| \right] \right) B_{0,d}^\dagger \otimes |\perp\rangle\langle\perp|^S \\
= &\sum_{\substack{(b,a)\neq(0^n,0^{2n}) \\ d}} B_{b,d}^{MT_1R} \left(\mathrm{Tr}_M \left[A_a \rho^{MR} A_a^\dagger \right] \otimes \tau^{MT_1} \right) B_{b,d}^\dagger \otimes |\perp\rangle\langle\perp|^S \\
= &\ \text{Eq. (21)} + \text{Eq. (22)}.
\end{aligned}$$

Tracing out register T_2 ensures that the second half of a and a' have to be equal; Twirling over the Clifford group ensures that the first half (acting on register T_1) of a and a' have to be equal (see the proof of Lemma A.1).

These derivations show that the output state that the environment sees (in registers MT_1RS) in the real protocol are negligibly close to the output state in the ideal protocol. This concludes our proof for the second case, where player 1 is dishonest. □

F Proof of Lemma 5.3

Proof. For the sake of clarity, assume again that there is only one wire, held by player 1 (who might be honest or dishonest). Generalizing the proof to multiple wires does not require any new

technical ingredients, but simply requires a lot more (cluttering) notation.

In the protocol $\Pi^{G^{m_\ell}} \diamond \mathfrak{I}^C$, an adversary $\mathcal{A}$ receives a state ρ^{MR} from the environment (where again, $M := R_1^{\mathsf{in}}$). It potentially alters this state with a unitary map A, submits the result to the ideal functionality, and receives the register $MT_1 = R_1^{\mathsf{out}}$. The adversary may again act on the state, say with a map B, and then gets a chance to submit (for all players $i \in I_{\mathcal{A}}$) bits b_i to $\mathfrak{I}^C$ (or $\Pi^{G^{m_\ell}}$). If one or more of those bits are 1, the ideal functionality (or the MPC) aborts by overwriting the state register S with $\perp$.

In case all bits are 0, the output register MT_1RS contains

$$\begin{aligned}&\mathop{\mathbb{E}}_{E} B^{MT_1R} E^{MT_1} \left(\mathcal{C}^M \left(A\rho^{MR}A^\dagger\right) \otimes |0^n\rangle\langle 0^n|^{T_1}\right) E^\dagger B^\dagger \otimes \left|E((G^{m_\ell})^\dagger \otimes \mathsf{I}^{\otimes n})\right\rangle\!\left\langle E((G^{m_\ell})^\dagger \otimes \mathsf{I}^{\otimes n})\right|^S \\ =&\mathop{\mathbb{E}}_{E} B^{MT_1R} E^{MT_1} (G^{m_\ell})^M \left(\mathcal{C}^M \left(A\rho^{MR}A^\dagger\right) \otimes |0^n\rangle\langle 0^n|^{T_1}\right) (G^{m_\ell})^\dagger E^\dagger B^\dagger \otimes |E\rangle\langle E|^S, \end{aligned} \tag{31}$$

where $\mathcal{C}(\cdot)$ is the map induced by the circuit C.

In case not all bits are 0, the output register MT_1RS contains

$$B'A'\rho_R(B'A')^\dagger \otimes \tau^{MT_1} \otimes |\perp\rangle\langle\perp|^S, \tag{32}$$

where A' and B' are the reduced maps A and B on register R.

Define a simulator $\mathcal{S}$ as follows:

Simulator 3. On input ρ^{MR} from the environment, do:

- Run A on MR.
- Submit M to the ideal functionality for $G^{m_\ell} \circ C$, and receive MT_1.
- Run B on MT_1R, and note its output bits (b_i, b_i') for all $i \in I_A$. Submit $\max\{b_i, b_i'\}$ to the ideal functionality for $G^{m_\ell} \circ C$.

From the point of view of the adversary, the state it receives from the ideal functionality is the same: a Clifford-encoded state. Thus, the bits b_i and b_i' will not be different in this simulated scenario. In fact, the output state is exactly Eq. (31) + Eq. (32). □

G Proof of Lemma 5.5

Proof. There are four different cases, for which we construct simulators separately: both players involved in the CNOT are honest ($i, j \notin I_{\mathcal{A}}$), both players are dishonest ($i, j \in I_{\mathcal{A}}$), only player i is honest ($i \notin I_{\mathcal{A}}$, $j \in I_{\mathcal{A}}$), or only player j is honest ($i \in I_{\mathcal{A}}$, $j \notin I_{\mathcal{A}}$). Without loss of generality, we will assume that all other players are dishonest (except in the second case, where at least one of the other players has to be honest), and that they have no inputs themselves: their encoded inputs can be regarded as part of the adversary's side information R. Note that these four cases also cover the possibility that $i = j$.

Case 1: player i and j are honest. In this case, the adversarial players in $I_{\mathcal{A}}$ only have influence on the execution of step 3 of the protocol, where the state is sent around in order for the players to jointly apply the random Clifford D.

As in the first case of the proof of Lemma 4.3 for the encoding protocol Π^{Enc} (where the encoding player is honest), define a simulator that performs a Pauli filter $\mathsf{IdFilter}$ on the attack of the adversary. The simulator and proof are almost identical to those in Lemma 4.3, so we omit the details here.

Case 2: player i and j are dishonest. Without loss of generality, we can break up the attack of the adversary (acting jointly for players i, j and any other players in $I_{\mathcal{A}}$) into three unitary operations, acting on the relevant register plus a side-information register. As in the proof of Lemma 4.3, we may assume that the honest actions are executed as well, since each attack may start or end with undoing that honest action. The first attack $A^{M^{ij}R}$ is executed on the plaintexts, before any protocol starts. The second attack $\tilde{A}^{M^{ij}T_{12}^{ij}R}$ happens after step 2 of Protocol 5.4, on the output of $\mathfrak{I}^C$ and the initialized registers T_2^{ij}. Finally, the third attack $\tilde{\tilde{A}}^{M^{ij}T_{12}^{ij}R}$ happens toward the end of the protocol, right before the T_2^{ij} registers are measured in step 6c of Protocol 5.4. Note that $\tilde{\tilde{A}}$ may depend on the instructions V, W_i and W_j.

It will be useful to decompose the second and third attacks as follows:

$$\tilde{A} = \sum_{a_1^i,a_2^i,a_1^j,a_2^j,c_1^i,c_2^i,c_1^j,c_2^j} (\mathsf{X}^{a_1^i}\mathsf{Z}^{c_1^i})^{M^iT_1^i} \otimes (\mathsf{X}^{a_2^i}\mathsf{Z}^{c_2^i})^{T_2^i} \otimes (\mathsf{X}^{a_1^j}\mathsf{Z}^{c_1^j})^{M^jT_1^j} \otimes (\mathsf{X}^{a_2^j}\mathsf{Z}^{c_2^j})^{T_2^j} \otimes \tilde{A}^R_{a_{12}^{ij},c_{12}^{ij}} \tag{33}$$

$$\tilde{\tilde{A}} = \sum_{b,d} (\mathsf{X}^b\mathsf{Z}^d)^{T_2^{ij}} \otimes \tilde{\tilde{A}}_{b,d}^{M^{ij}T_1^{ij}R} \tag{34}$$

Whenever the order is clear from the context, we will abbreviate, for example, a_{12}^{ij} for the concatenation $a_1^ia_2^ia_1^ja_2^j$, as we have done in the last term of Equation (33).

In terms of an arbitrary attack $A, \tilde{A}, \tilde{\tilde{A}}$, define the simulator $\mathcal{S}$ as follows:

Simulator 4. On input $\rho^{M^iM^jR}$ from the environment, do:

1. Initialize $b_i = 0$.
2. Run A on $M^{ij}R$.
3. Submit M^{ij} to $\mathfrak{I}^{\mathsf{CNOT}^{m_\ell}\circ C}$, and receive $M^{ij}T_1^{ij}$, containing an encodings of the M^i and M^j registers of $\mathsf{CNOT}^{m_\ell}(C(\rho))$, under some (secret) keys E_i, E_j.
4. Run $\mathsf{ZeroFilter}^{T_2^{ij}}(\mathsf{IdFilter}^{M^{ij}T_1^{ij}}(\tilde{A}))$ on R (see Section 2.3). If one of the filter flags is 1, set $b_i = 1$.
5. Sample random $V' \in \mathscr{C}_{4n+2}$ and $W_i', W_j' \in \mathscr{C}_{2n+1}$, and run $\mathsf{XFilter}^{T_2^{ij}}(\tilde{\tilde{A}})$ on $M^{ij}T_1^{ij}R$, where $\tilde{\tilde{A}}$ may depend on V', W_i', W_j'. If the filter flag is 1, set $b_i = 1$.
6. Submit b_i to the ideal functionality, along with all other $b_\ell = 0$ for $\ell \in I_{\mathcal{A}} \setminus \{b_i\}$.

The simulator should also abort whenever the adversary signals abort during an interaction with MPC. For simplicity, we leave out these abort bits in the simulator and proof. They are dealt with in the same way as in the proof of Lemma 5.3.

As before, we derive the real and ideal output states in the registers $R_i^{\mathsf{out}} = M^i T_1^i$ and $R_j^{\mathsf{out}} = M^j T_i^j$, the state register S, and the attacker's side information R, and aim to show that they are negligibly close in terms of the security parameter n.

In the ideal (simulated) case, there are two points at which cheating may be detected by the simulator: once during the zero/identity filter of $\tilde{A}$, and during the X filter of $\tilde{\tilde{A}}$. Thus, there are three possible outcome scenarios: both tests are passed, the first test is passed but the second is not, or the first test fails (in which case it does not matter whether the second test is passed or not).

If both tests pass, then by three applications of Lemma 2.3, the simulated output state is

$$\mathop{\mathbb{E}}_{E_i,E_j} \tilde{\tilde{A}}_0^{M^{ij}T_1^{ij}R} \tilde{A}_{0,0}^{R} \left(E_i \otimes E_j\right)^{M^{ij}T_1^{ij}} \left(\mathsf{CNOT}^{m_\ell} \mathcal{C}\left(A\rho A^\dagger\right) \mathsf{CNOT}^{m_\ell \dagger} \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_1^{ij}}\right) \left(E_i \otimes E_j\right)^\dagger \tilde{A}_{0,0}^\dagger \tilde{\tilde{A}}_0^\dagger \otimes \left|E_i, E_j\right\rangle\!\left\langle E_i, E_j\right|^S, \tag{35}$$

where we write $\tilde{A}_{0,0}$ to denote the attack $\sum_{c_2^{ij}} \tilde{A}_{0000,0c_2^i 0c_2^j}$ that passes through the zero/identity filter, and $\tilde{\tilde{A}}_0$ to denote the attack $\sum_d \tilde{\tilde{A}}_{0,d}$ that passes through the X filter.

If the first test is passed but the second test is not, then the storage register S gets erased, so that we may view the E_i and E_j operations as Clifford twirls of the registers they encode. In that case, the (simulated) output state is

$$\sum_{b\neq 0} \tilde{\tilde{A}}_b \tilde{A}_{0,0} \mathcal{T}_{\mathscr{C}_{n+1}}^{M^i T_1^i} \left(\mathcal{T}_{\mathscr{C}_{n+1}}^{M^j T_1^j} \left(\mathsf{CNOT}^{m_\ell} \mathcal{C}\left(A\rho A^\dagger\right) \mathsf{CNOT}^{m_\ell \dagger} \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|\right)\right) \tilde{A}_{0,0}^\dagger \tilde{\tilde{A}}_b^\dagger \otimes \left|\perp\right\rangle\!\left\langle\perp\right|^S \tag{36}$$

$$= \sum_{b\neq 0} \tilde{\tilde{A}}_b \tilde{A}_{0,0} \left(\mathrm{Tr}_{M^{ij}}\left[A\rho^{M^{ij}R}A^\dagger\right] \otimes \tau^{MT_1^{ij}}\right) \tilde{A}_{0,0}^\dagger \tilde{\tilde{A}}_b^\dagger \otimes \left|\perp\right\rangle\!\left\langle\perp\right|^S. \tag{37}$$

The Clifford twirls cause the data and trap registers to become fully mixed, thereby also nullifying the effect of the CNOT and circuit C on the data.

Finally, we consider the third scenario, where already the first test (the zero / identity filter) fails. As in the previous scenario, the storage register S is erased, allowing us to apply the Clifford twirl again. By Lemma 2.3, the output state in this case is

$$\sum_b \sum_{\substack{(a_{12}^{ij}, c_1^{ij}) \neq \\ (0^{4n+2}, 0^{2n+2})}} \tilde{\tilde{A}}_b \tilde{A}_{a_{12}^{ij}, c_1^{ij}} \left(\mathrm{Tr}_{M^{ij}}\left[A\rho^{M^{ij}R}A^\dagger\right] \otimes \tau^{MT_1^{ij}}\right) \tilde{A}_{a_{12}^{ij}, c_1^{ij}}^\dagger \tilde{\tilde{A}}_b^\dagger \otimes \left|\perp\right\rangle\!\left\langle\perp\right|^S, \tag{38}$$

writing $\tilde{A}_{a_{12}^{ij}, c_1^{ij}} := \sum_{c_2^{ij}} \tilde{A}_{a_{12}^{ij}, c_{12}^{ij}}$. Note that for the second test (the X filter), the terms for both possible flag values remain: the cheating bit b_i is already set to 1, regardless of the outcome of this second test.

We move on to the analysis of the real protocol $\Pi^{\mathsf{CNOT}^{m_\ell}} \diamond \mathfrak{I}^C$, and aim to show that the output state is equal to Eq. (35) + Eq. (37) + Eq. (38). To do so, consider the output state of the real protocol, right before the final measurement.

We continue to argue why the attacks are independent of E_{ij}, E'_{ij}, g_{ij}, r_{ij} and s_{ij}. The intuition for this fact is that D is uniformly random and independent of E_{ij} from the perspective of the

adversary. Therefore it "hides" all other information that is used to compile V, including F_{ij}. Therefore F_{ij} are as random and independent as D from the perspective of the adversary, i.e. given V. This allows for a similar argument for the Cliffords W_{ij}, where now F hides all the other information, i.e. E'_{ij}, g_{ij}, r_{ij} and s_{ij}.

For the following more formal argument, we treat all the mentioned quantities as random variables. Initially, E_{ij} are uniformly random. D is the product of a number of Clifford group elements, at least one of which is generated honestly and therefore sampled uniformly at random. But for any group G, given two independent random variables ζ and η on G, where ζ is uniformly random, we have that $\eta\zeta$ is uniformly random and $\eta\zeta \perp \eta$, where $\perp$ denotes independence. This implies that D is indeed a uniformly random Clifford itself. Using the same argument, V is uniformly random and $V \perp (E_{ij}, F_{ij})$. The quantities E'_{ij}, g_{ij}, r_{ij} and s_{ij} are sampled independently and uniformly after V is handed to player i, so we even have $V \perp (E_{ij}, F_{ij}, E'_{ij}, g_{ij}, r_{ij}, s_{ij})$. After step 4. in Protocol 5.4, the adversary has a description of V, so when analyzing W_{ij}, we have to derive independence statements *given* V. But as shown before F_{ij} are independent of V, so the the group random variable property above we have $W_{ij} \perp (E'_{ij}, g_{ij}, r_{ij}, s_{ij}) | V$. Clearly, E_{ij} is independent of all the random variables used in W_{ij}, and we have shown that $E_{ij} \perp V$, so $W_{ij} \perp (E_{ij}, E'_{ij}, g_{ij}, r_{ij}, s_{ij}) | V$. In summary, we have

$$(V, W_{ij}) \perp (E_{ij}, E'_{ij}, g_{ij}, r_{ij}, s_{ij}). \tag{39}$$

According to the decomposition of the attack into attack maps $A, \tilde{A}$ and $\tilde{\tilde{A}}$, that we made without loss of generality, the Clifford operations F_i, F_j, and D cancel again after having fulfilled their task of hiding information, which allows us to utilize Equation (39) to carry out the expectation values over various variables from the right hand side of that equation.

The output state of the real protocol is

$$\mathop{\mathbb{E}}_{\substack{E'_i, E'_j, g_i, g_j \\ r_i, s_i, r_j, s_j}} \tilde{\tilde{A}}^{M^{ij} T_{12}^{ij} R} \left(E'_i \otimes (\mathsf{X}^{r_i}\mathsf{Z}^{s_i})^{T_2^i} \otimes E'_j \otimes (\mathsf{X}^{r_j}\mathsf{Z}^{s_j})^{T_2^j} \right) \left(U_{g_i}^{T_{12}^i} \otimes U_{g_j}^{T_{12}^j} \right) \mathsf{CNOT}^{m_\ell} \sigma \, \mathsf{CNOT}^{m_\ell \dagger}$$
$$\left(U_{g_i}^\dagger \otimes U_{g_j}^\dagger \right) \left(E'_i \otimes (\mathsf{X}^{r_i}\mathsf{Z}^{s_i}) \otimes E'_j \otimes (\mathsf{X}^{r_j}\mathsf{Z}^{s_j}) \right)^\dagger \tilde{\tilde{A}}^\dagger \otimes \left| E'_i, E'_j \right\rangle \left\langle E'_i, E'_j \right|^S, \tag{40}$$

where (again writing $\mathcal{C}(\cdot)$ for the map induced by the circuit C)

$$
\begin{aligned}
\sigma := & \mathop{\mathbb{E}}_{E_i,E_j\in\mathscr{C}_{n+1}} \left(E_i^\dagger \otimes E_j^\dagger\right) \tilde{A}^{M^{ij}T_{12}^{ij}R} \Bigg(\left(E_i^{M^iT_1^i} \otimes E_j^{M^jT_1^j}\right) \left(\mathcal{C}\left(A\rho^{M^{ij}R}A^\dagger\right) \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_1^{ij}}\right) \\
& \qquad \left(E_i^\dagger \otimes E_j^\dagger\right) \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_2^{ij}} \Bigg) \tilde{A}^\dagger \left(E_i \otimes E_j\right) \\
= & \, \mathcal{T}_{\mathscr{C}_{n+1}}^{M^iT_1^i}\left(\mathcal{T}_{\mathscr{C}_{n+1}}^{M^jT_1^j}\left(\tilde{A}\right)\right)\left(\mathcal{C}\left(A\rho^{M^{ij}R}A^\dagger\right) \otimes \left|0^{4n}\right\rangle\!\left\langle 0^{4n}\right|^{T_{12}^{ij}}\right) \\
\approx_{\mathsf{negl}(n)} & \, \tilde{A}_{00}^{T_2^{ij}R}\left(\mathcal{C}\left(A\rho^{M^{ij}R}A^\dagger\right) \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_2^{ij}}\right)\tilde{A}_{00}^\dagger \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_1^{ij}} \\
& + \mathrm{Tr}_{M^i}\left[\tilde{A}_{01}^{T_2^{ij}R}\left(\mathcal{C}\left(A\rho^{M^{ij}R}A^\dagger\right) \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_2^{ij}}\right)\tilde{A}_{01}^\dagger\right] \otimes \tau^{M^iT_1^i} \otimes \left|0^n\right\rangle\!\left\langle 0^n\right|^{T_1^j} \\
& + \mathrm{Tr}_{M^j}\left[\tilde{A}_{10}^{T_2^{ij}R}\left(\mathcal{C}\left(A\rho^{M^{ij}R}A^\dagger\right) \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_2^{ij}}\right)\tilde{A}_{10}^\dagger\right] \otimes \left|0^n\right\rangle\!\left\langle 0^n\right|^{T_1^i} \otimes \tau^{M^jT_1^j} \\
& + \mathrm{Tr}_{M^{ij}}\left[\tilde{A}_{11}^{T_2^{ij}R}\left(\mathcal{C}\left(A\rho^{M^{ij}R}A^\dagger\right) \otimes \left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|^{T_2^{ij}}\right)\tilde{A}_{11}^\dagger\right] \otimes \tau^{M^{ij}T_1^{ij}},
\end{aligned} \tag{41}
$$

and

$$
\tilde{A}_{pq} := \sum_{\substack{a_2^{ij},c_2^{ij} \\ a_1^ic_1^i\in S_p \\ a_1^jc_1^j\in S_q}} (\mathsf{X}^{a_2^{ij}}\mathsf{Z}^{c_2^{ij}})^{T_2^{ij}} \otimes \tilde{A}^R_{a_{12}^ia_{12}^j,c_{12}^ic_{12}^j}.
$$

for $p, q \in \{0,1\}$ and $S_0 := \{0^{2n+2}\}$, $S_1 := \{0,1\}^{2n+2} \setminus S_0$. The approximation follows by a double application of Lemma C.1. We can twirl with the keys E_i and E_j, since none of the attacks can depend on the secret encoding keys E_i, E_j, and the keys have been removed from the storage register S, and replaced by the new keys E'_i, E'_j.

Having rewritten the state σ in this form, we consider the state in Equation (40) after the T_2^{ij} registers are measured in the computational basis, as in step 6c of Protocol 5.4. We first consider the case where the measurement outcome is accepted by the MPC (i.e., the measurement outcome is r_ir_j). Using the same derivation steps as in Equations (23)–(30), we see that the real accept state approximates (up to a negligible error in n)

$$
\begin{aligned}
\mathop{\mathbb{E}}_{E'_i,E'_j\in\mathscr{C}_{n+1}} & \tilde{\tilde{A}}_0^{M^{ij}T_1^{ij}R}\left(E'_i \otimes E'_j\right)^{M^{ij}T_1^{ij}} \mathrm{Tr}_{T_2^{ij}}\left[\left|0^{4n}\right\rangle\!\left\langle 0^{4n}\right|^{T_{12}^{ij}} \mathsf{CNOT}^{m_\ell}\sigma\mathsf{CNOT}^{m_\ell\dagger}\left|0^{4n}\right\rangle\!\left\langle 0^{4n}\right|\right] \\
& \left(E'_i \otimes E'_j\right)^\dagger \tilde{\tilde{A}}_0^\dagger \otimes \left|E'_i,E'_j\right\rangle\!\left\langle E'_i,E'_j\right|^S.
\end{aligned} \tag{42}
$$

To derive the above expression, we applied a Pauli twirl, which relies on the fact that the adversary cannot learn r_i, r_j, s_i, s_j. Furthermore, the derivation contains an application of Lemma E.1 to expand the effect of measuring T_2^{ij} to measuring both registers T_{12}^{ij}. To apply this lemma, we use the aforementioned fact that g_i and g_j remain hidden from the adversary.

The second, third, and fourth terms of the sum in the approximation of σ (see Equation (41)) have negligible weight inside Equation (42), since the probability of measuring an all-zero string in

the T_1^{ij} registers is negligible in n whenever one or both are in the fully mixed state τ. Additionally, the only components in $\tilde{A}_{00}$ that survive are those that act trivially in the computational basis on T_2^{ij}. Hence,

$$\text{Eq. (42)} \approx_{\mathsf{negl}(n)} \text{Eq. (35)}.$$

In case the measurement outcome is rejected by the MPC (i.e., it is anything other than $r_i r_j$), the output state can be derived using the same steps that were used to obtain Equation (30) in the proof of Lemma 4.3. Up to an error negligible in n, it approximates

$$\sum_b \tilde{\tilde{A}}_b^{M^{ij}T_1^{ij}R} \mathcal{T}_{\mathscr{C}_{n+1}}^{M^iT_1^i}\left(\mathcal{T}_{\mathscr{C}_{n+1}}^{M^jT_1^j}\left(\mathrm{Tr}_{T_2^{ij}}\left[(\mathbb{I}-\Pi_{b,F})^{T_{12}^{ij}}\mathsf{CNOT}^{m_\ell}\sigma\mathsf{CNOT}^{m_\ell\dagger}(\mathbb{I}-\Pi_{b,F})^\dagger\right]\right)\right)\tilde{\tilde{A}}_b^\dagger \otimes |\perp\rangle\langle\perp|^S .$$

The encoding under the keys E_i', E_j' in Equation (40) can be regarded as two Clifford twirls, because these keys are removed from the storage register S, and because the attack maps also cannot depend on them, since they are unknown by the adversary.

The next step is to substitute the expression for σ that was derived in Equation (41). We distinguish between the case $b \neq 0$, where $\mathbb{I} - \Pi_{b,F} = \mathbb{I}$ and thus all terms of Equation (41) remain, and the case $b = 0$, where one has to more carefully count which (parts of the) terms remain. To do so, observe that the first term is projected to non-zero in T_{12}^{ij} whenever ${a_2}^{ij}$ is nonzero. The other three terms are always projected to non-zero, up to a negligible contribution of the all-zero string in the fully mixed state τ. In summary, exactly those terms $\tilde{A}^R_{a_{12}^{ij},c_{12}^{ij}}$ remain for which $(a_{12}^{ij}, c_1^{ij}) \neq (0^{4n+2}, 0^{2n+1})$.

Because the two Clifford twirls map the M^{ij} registers to a fully mixed state, the four terms in Equation (41) can be combined, resulting in the following output state in the reject case

$$\begin{aligned}
&\sum_{b\neq 0}\sum_{a_{12}^{ij},c_1^{ij}} \tilde{\tilde{A}}_b \tilde{A}_{a_{12}^{ij},c_1^{ij}}\left(\mathrm{Tr}_{M^{ij}}\left[A\rho^{M^{ij}R}A^\dagger\right]\otimes\tau^{MT_1^{ij}}\right)\tilde{A}^\dagger_{a_{12}^{ij},c_1^{ij}}\tilde{\tilde{A}}_b^\dagger\otimes|\perp\rangle\langle\perp|^S \\
+ &\sum_{\substack{(a_{12}^{ij},c_1^{ij})\neq\\(0^{4n+2},0^{2n+2})}} \tilde{\tilde{A}}_0 \tilde{A}_{a_{12}^{ij},c_1^{ij}}\left(\mathrm{Tr}_{M^{ij}}\left[A\rho^{M^{ij}R}A^\dagger\right]\otimes\tau^{MT_1^{ij}}\right)\tilde{A}^\dagger_{a_{12}^{ij},c_1^{ij}}\tilde{\tilde{A}}_0^\dagger\otimes|\perp\rangle\langle\perp|^S \\
&\qquad = \text{Eq. (37)} + \text{Eq. (38)}.
\end{aligned}$$

We have shown that the sum of the three terms of the output state in the simulated case (both tests accept, the first test accepts but the second rejects, and the first test rejects) is approximately equal to the sum of the two terms of the output state in the real case (the MPC accepts the measurement outcome, or the MPC rejects the measurement outcome).

Case 3: only player i is honest. At first, it may seem that this is just a special case of the previous one, where both players are dishonest. While this is true in spirit, we cannot directly use the simulator from the previous case. The reason is syntactical: a simulator would not have access to the registers $M^iT_{12}^i$, because they are held by honest player i. Thus, the simulator needs to differ slightly from the previous case. However, it is very similar, as is the derivation of the real/ideal output states. We therefore omit the full proof, and instead only define the simulator.

The adversary again has three opportunities to attack: an attack A on the plaintext and side-information register M^jR, which happens before the ideal functionality $\mathfrak{I}^C$ is called; an attack $\tilde{A}$

on the output of $\mathfrak{I}^C$ in registers $M^j T_1^j R$ (right before player j sends their state to player i); and an attack $\tilde{\tilde{A}}$ on $M^j T_{12}^j R$, after an honest application of W_j (which we may assume to happen without loss of generality), but before the computational-basis measurement of T_2. Given these attacks, define the simulator as follows.

Simulator 5. On input $\rho^{M^j R}$ from the environment, do:

1. Initialize $b_j = 0$.
2. Run A on $M^j R$.
3. Submit M^j to the ideal functionality $\mathfrak{I}^{\mathsf{CNOT}^{m_\ell} \circ C}$, and receive $M^j T_1^j$, containing an encoding under a secret key E_j. (Honest player i holds the other output, encoded under E_i.)
4. Run $\mathsf{IdFilter}^{M^j T_1^j}(\tilde{A})$ on R. If the filter flag is 1, then set $b_j = 1$.
5. Sample random $W'_j \in \mathscr{C}_{2n+1}$, and run $\mathsf{XFilter}^{T_2^j}(\tilde{\tilde{A}})$ on $M^j T_1^j R$, where $\tilde{\tilde{A}}$ may depend on W'_j. If the filter flag is 1, then set $b_j = 1$.
6. Submit b_j to the ideal functionality, along with all other $b_\ell = 0$ for $\ell \in I_{\mathcal{A}} \setminus \{b_j\}$.

Intuitively, the simulator tests whether player j sent the actual outcome of $\mathfrak{I}^C$ without altering it (step 4 of the simulator), and whether player j left the computational basis of T_2 invariant before measuring it (step 5 of the simulator).

Case 4: only player j is honest. Similarly to the previous case, we need to provide a separate simulator for the case where player i is dishonest, player j is honest, and (without loss of generality) all other players are dishonest.

The adversary has three opportunities to attack: an attack A on the plaintext and side-information register $M^i R$, which happens before the ideal functionality $\mathfrak{I}^C$ is called; an attack $\tilde{A}$ on registers $M^{ij} T_{12}^{ij} R$ that is applied on the outputs of the ideal functionality and on the extra registers T_2, right before D is applied; and an attack $\tilde{\tilde{A}}$ on $M^{ij} T_{12}^{ij} R$, right before the measurement on T_2^i (as part of player i's test) and the application of W_j (so right before sending the appropriate registers to player j). Given these attacks, define the simulator as follows.

Simulator 6. On input $\rho^{M^i R}$ from the environment, do:

1. Initialize $b_i = 0$.
2. Run A on $M^i R$.
3. Submit M^i to the ideal functionality $\mathfrak{I}^{\mathsf{CNOT}^{m_\ell} \circ C}$, and receive $M^i T_1^i$, containing an encoding under a secret key E_i. (Honest player j holds the other output, encoded under E_j.)

4. Run $\mathsf{ZeroFilter}^{T_2^{ij}}\left(\mathsf{IdFilter}^{M^{ij}T_1^{ij}}(\tilde{A})\right)$ on R. If the filter flag is 1, then set $b_i = 1$.

5. Sample random $V, W_i' \in \mathscr{C}_{2n+1}$, and run $\mathsf{XFilter}^{T_2^i}\left(\mathsf{IdFilter}^{M^jT_{12}^j}(\tilde{\tilde{A}})\right)$ on $M^iT_1^jR$, where $\tilde{\tilde{A}}$ may depend on V and W_i'. If the filter flag is 1, then set $b_i = 1$.

6. Submit b_i to the ideal functionality, along with all other $b_\ell = 0$ for $\ell \in I_{\mathcal{A}} \setminus \{b_i\}$.

Intuitively, the simulator tests (in step 4) whether player i leaves the states received from the ideal functionality and player j intact, as well as the traps in T_2^{ij} that are initialized to $\left|0^{2n}\right\rangle\!\left\langle 0^{2n}\right|$. In step 5, it tests both whether player i executes the test honestly by not altering the computational-basis value of T_2^i, and whether he would give the correct (uncorrupted) state to player j.

□

H Proof of Lemma 5.7

The following lemma captures the fact that $\mathsf{CNOT}_{1,c}$ makes it hard to alter the outcome of a computational-basis measurement with a (Pauli) attack X^b if b does not depend on c. We will use this lemma later in the security proof of the measurement protocol.

Lemma H.1. *Let $m \in \{0,1\}$, and let ρ be a single-qubit state. Let $p : \{0,1\}^{n+1} \to [0,1]$ be a probability distribution, and u_n the uniform distribution on $\{0,1\}^n$. Then*

$$\left\| \mathop{\mathbb{E}}_{\substack{b\sim p\\ c\sim u_n}} \langle m, m\cdot c|\, \mathsf{X}^b\mathsf{CNOT}_{1,c}\left(\rho \otimes |0^n\rangle\!\langle 0^n|\right)\mathsf{CNOT}_{1,c}^\dagger \mathsf{X}^b\, |m, m\cdot c\rangle - p(0^{n+1})\,\langle m|\,\rho\,|m\rangle \right\|_1 \leqslant 2^{-n}.$$

Proof. By commutation relations between CNOT and X, we have that for all b and c,

$$\mathsf{X}^b\mathsf{CNOT}_{1,c} = \mathsf{CNOT}_{1,c}\mathsf{X}^{b\oplus(0,b_1\cdot c)},$$

where b_1 denotes the first bit of b. Furthermore, $\mathsf{CNOT}_{1,c}\,|m, m\cdot c\rangle = |m, 0^n\rangle$. Using these two equalities, we have

$$\begin{aligned} &\mathop{\mathbb{E}}_{\substack{b\sim p\\ c\sim u_n}} \langle m, m\cdot c|\, \mathsf{X}^b\mathsf{CNOT}_{1,c}\left(\rho \otimes |0^n\rangle\!\langle 0^n|\right)\mathsf{CNOT}_{1,c}^\dagger \mathsf{X}^b\, |m, m\cdot c\rangle \\ =\ &\mathop{\mathbb{E}}_{\substack{b\sim p\\ c\sim u_n}} \langle m, 0^n|\, \mathsf{X}^{b\oplus(0,b_1\cdot c)}\left(\rho \otimes |0^n\rangle\!\langle 0^n|\right)\mathsf{X}^{b\oplus(0,b_1\cdot c)}\, |m, 0^n\rangle\,. \end{aligned} \tag{43}$$

Let us consider which values of b result in a non-zero term. In order for the last n qubits to be in the $|0^n\rangle\!\langle 0^n|$ state after $\mathsf{X}^{b\oplus(0,b_1\cdot c)}$, it is necessary that $b\oplus(0,b_1\cdot c) \in \{(0,0^n),(1,0^n)\}$. By considering the two possible cases $b_1 = 0$ and $b_1 = 1$, we see that the only two values of b for which this is the case are $b = (0,0^n)$ and $b = (1,c)$. Thus Equation (43) equals

$$\begin{aligned} &\mathop{\mathbb{E}}_{c\sim u_n} p(0^{n+1})\,\langle m, 0^n|\left(\rho\otimes|0^n\rangle\!\langle 0^n|\right)|m,0^n\rangle && +p(c)\,\langle m,0^n|\,\mathsf{X}^{1,0^n}\left(\rho\otimes|0^n\rangle\!\langle 0^n|\right)\mathsf{X}^{1,0^n}\,|m,0^n\rangle \\ =\ & p(0^{n+1})\,\langle m|\,\rho\,|m\rangle && + \mathop{\mathbb{E}}_{c\sim u_n} p(c)\,\langle m+1|\,\rho\,|m+1\rangle \\ \approx_{2^{-n}}\ & p(0^{n+1})\,\langle m|\,\rho\,|m\rangle\,. \end{aligned}$$

The last step follows from the fact that $\mathbb{E}_c\, p(c) = 2^{-n}$. □

We now move on to proving the security of Protocol 5.6 by showing that its outcome resembles that of the ideal functionality.

of Lemma 5.7. Let player i be the player holding (the encoding of) the state in wire w (assume, for simplicity, that w is the only wire in the computation). If player i is honest, then it is simple to check that the outcome is correct: the unitary V is designed so that, whatever the first (data) qubit collapses to, all other qubits that appear in s measure to the same value. In step 4, the MPC checks that this is indeed the case, and stores the measured value in the state register.

For the rest of this proof, we will assume that player i is dishonest. The other players do not play a role, except for their power to abort the ideal functionalities and/or MPC. We do not fix which players in $[k] \setminus \{i\}$ are honest: as long as at least one of them is, the encoding key E will be unknown to the adversary.

In an execution of $\Pi^{\curlywedge} \diamond \mathfrak{I}^C$, an adversary has two opportunities to influence the outcome: before and after interacting with the ideal functionality for C. Before the adversary submits the register $M = R_i^{\mathsf{in}}$ to $\mathfrak{I}^C$, it applies an arbitrary attack unitary A to the register MR it receives from the environment. (Recall that R is a side-information register.) Afterwards, it can act on $MT_1 = R_i^{\mathsf{out}}$ and R, and produces two bits (b_i to signal cheating to $\mathfrak{I}^C$, and b_i' to signal cheating to the MPC which is part of $\Pi^{\curlywedge}$), plus a bit string. We may assume, without loss of generality, that the adversary first applies the honest unitary V, followed by an arbitrary (unitary) attack B and subsequently by an honest computational-basis measurement of the registers MT_1.

For any adversary, specified by the unitaries A and B, define a simulator $\mathcal{S}$ as follows:

Simulator 7. On input ρ^{MR} from the environment, do:

1. Run A on registers MR.
2. Sample a random $F \in \mathscr{C}_{n+1}$ and a random $r \in \{0,1\}^{n+1}$.
3. Prepare the state $F\left|r\right\rangle\!\left\langle r\right|F^\dagger$ in a separate register XT_1, and apply the map B to XT_1R, using the instruction $F^\dagger$ instead of T.
4. Measure XT_1 in the computational basis, and check that the outcome is r. If so, submit M to $\mathfrak{I}^{\curlywedge \circ C}$, along with a bit $b = 0$ (no cheating). Otherwise, submit M and $b = 1$.

Throughout this proof, we decompose the attack B as

$$B = \sum_{b,d \in \{0,1\}^{n+1}} \left(\mathsf{X}^b\mathsf{Z}^d\right)^{MT_1} \otimes B_{b,d}^R, \tag{44}$$

and similarly as before, we abbreviate $B_b := \sum_d B_{b,d}$ (and B_0 for $B_{0^{n+1}}$).

We analyze the output state in registers RS (note that the MT_1 registers are destroyed by the measurement) in both the ideal and the real case, and aim to show that they are indistinguishable, whatever the input ρ^{MR} was.

In the ideal (simulated) case, first consider the output state in case of accept. Write $\mathcal{C}(\cdot)$ for the map induced by the circuit C. Following the steps of the simulator, abbreviating $\sigma = A^{MR}\rho^{MR}A^\dagger$, and decomposing B as in Equation (44), we see that the output in RS in case of accept is

$$\sum_{m\in\{0,1\}^m} \mathop{\mathbb{E}}_r \langle m|^M \mathcal{C}^M \left(\langle r|^{XT_1} B^{XT_1R} \left(\sigma \otimes \left(F^\dagger F |r\rangle\langle r| F^\dagger F\right)^{XT_1}\right) B^\dagger |r\rangle\right) |m\rangle \otimes |m\rangle\langle m|^S$$
$$= \sum_{m\in\{0,1\}} \sum_{b,d,b',d'} \mathop{\mathbb{E}}_r \langle m|^M \left(\mathcal{C}^M \left(B^R_{b,d}\sigma B^\dagger_{b',d'}\right) \otimes \langle r| \mathsf{X}^b\mathsf{Z}^d |r\rangle\langle r| \mathsf{Z}^{d'}\mathsf{X}^{b'} |r\rangle^{XT_1}\right) |m\rangle \otimes |m\rangle\langle m|^S$$
$$= \sum_{m\in\{0,1\}} \langle m|^M \mathcal{C}^M \left(B^R_0 A^{MR}\rho^{MR}A^\dagger B^\dagger_0\right) |m\rangle \otimes |m\rangle\langle m|^S . \tag{45}$$

The ideal reject case is similar, except we project onto $\mathbb{I} - |r\rangle\langle r|$ instead of onto $|r\rangle\langle r|$. The output state is

$$\sum_{m\in\{0,1\}} \sum_{b\neq 0^{n+1}} \langle m|^M \mathcal{C}^M \left(B^R_b A^{MR}\rho^{MR}A^\dagger B^\dagger_b\right) |m\rangle \otimes |\perp\rangle\langle\perp|^S$$
$$= \sum_{b\neq 0^{n+1}} Tr_M \left[B^R_b A^{MR}\rho^{MR}A^\dagger B^\dagger_b\right] \otimes |\perp\rangle\langle\perp|^S . \tag{46}$$

In the real protocol, the unitary T does not reveal any information about c, so the attack B is independent of it. This allows us to apply Lemma H.1, after performing a Pauli twirl to decompose the attack B. Again abbreviating $\sigma = A\rho A^\dagger$, the state in the accept case is

$$= \mathop{\mathbb{E}}_c \sum_m \langle r \oplus (m, m\cdot c)|^{MT_1} B^{MT_1R}\mathsf{X}^r\mathsf{Z}^s\mathsf{CNOT}_{1,c}E^\dagger E \left(\mathcal{C}^M(\sigma) \otimes |0^n\rangle\langle 0^n|^{T_1}\right)$$
$$E^\dagger E\mathsf{CNOT}^\dagger_{1,c}\mathsf{Z}^s\mathsf{X}^r B^\dagger |r \oplus (m, m\cdot c)\rangle \otimes |m\rangle\langle m|^S$$
$$= \mathop{\mathbb{E}}_c \sum_{m,b} \langle m, m\cdot c| \mathsf{X}^b\mathsf{CNOT}_{1,c} \left(\mathcal{C}^M \left(B^R_b\sigma B^\dagger_b\right) \otimes |0^n\rangle\langle 0^n|\right) \mathsf{CNOT}^\dagger_{1,c}\mathsf{X}^b |m, m\cdot c\rangle \otimes |m\rangle\langle m|^S$$
$\approx_{2^{-n}}$ Eq. (45).

For the last step, observe that the probabilities $p(b)$ in the statement of Lemma H.1 are part of B_b.

Similarly, the real reject state is

$$\mathop{\mathbb{E}}_c \sum_{m,b} \sum_{x\neq(m,m\cdot c)} \langle x|^{MT_1} \mathsf{X}^b\mathsf{CNOT}_{1,c} \left(\mathcal{C}^M \left(B^R_b\sigma B^\dagger_b\right) \otimes |0^n\rangle\langle 0^n|^{T_1}\right) \mathsf{CNOT}^\dagger_{1,c}\mathsf{X}^b |x\rangle \otimes |\perp\rangle\langle\perp|^S$$
$\approx_{2^{-n}}$ Eq. (46).

In summary, we have shown that the output state in the real case is close to Eq. (45) + Eq. (46), for any input state ρ^{MR} provided by the environment $\mathcal{E}$. □

I Proof of Lemma 6.3

Before proving Lemma 6.3, we discuss the task of sampling in the quantum world.

Classically, some properties of a bit-string can be estimated just by querying a small fraction of it. For instance, in order to estimate the Hamming weight w of a n-bit string x, one could calculate

the Hamming weight w_S of a subset S of the bits of x and we have that $w \in [w_S - \delta, w_S + \delta]$ except with probability $O(2^{-2\delta^2|S|})$.

Such a result does not follow directly in the quantum setting, since the tested quantum state could, for instance, be entangled with the environment. However, Bouman and Fehr [BF10] have studied this problem in the quantum setting, and they showed that such sampling arguments also hold in the quantum setting, but with a quadratic loss in the error probability. A corollary of their result that will be important in this work is the following.

Lemma I.1 (Application of Theorem 3 of [BF10])**.** *Let $|\phi_{AE}\rangle \in (\mathbb{C}^2)^{\otimes n} \otimes \mathcal{H}_E$ be a quantum state and let $B = \{|v_0\rangle, |v_1\rangle\}$ be a fixed single-qubit basis. If we measure k random qubits of $Tr_E(|\phi_{AE}\rangle\langle\phi_{AE}|)$ in the B-basis and all of the outcomes are $|v_0\rangle$, then with probability $O(2^{-\delta^2 k})$, we have that*

$$|\phi_{AE}\rangle \in span\left((P_\pi |v_0\rangle^{\otimes n-t} |v_1\rangle^{\otimes t}) \otimes |\psi\rangle : 0 \leqslant t \leqslant \delta n, \pi \in S_n, |\psi\rangle \in \mathcal{H}_E\right).$$

We now proceed to the proof of Lemma 6.3.

of Lemma 6.3. The simulator for Π^{MS} is similar to the composed simulator for $\Pi^{\mathsf{Dec}} \diamond \Pi^{C} \diamond \Pi^{\mathsf{Enc}}$, where C is the Clifford circuit of Protocol 2.8. The difference is that the input is now chosen by player 1 instead of being given by the environment, and that each player tests if the decoded qubit is correct. We make a small modification for each of the following cases:

Case 1: player 1 is honest. In this case, the simulator only needs to also set $b_i = 1$ whenever the adversary aborts after it receives the output of the ideal quantum computation in Step 3 of Protocol 6.1. Otherwise, the simulator is exactly the same as the composed one.

Case 2: player 1 is dishonest. In this case, the simulator also tests if the decoded qubits by the (simulated) honest players in $[k] \setminus I_{\mathcal{A}}$ are indeed magic states of the correct form. More concretely, the simulator also measures all the qubits that the simulated players receive in the $\{|\mathsf{T}\rangle, |\mathsf{T}^\perp\rangle\}$ basis, and sets $b_i = 1$ if any of the outcomes is $|\mathsf{T}^\perp\rangle$. Otherwise, the simulator replaces the qubits in $[\ell]\setminus\left(\bigcup_{2\leqslant i\leqslant k} S_i\right)$ by true magic-states $|\mathsf{T}\rangle$, re-encodes them, and continues the composed simulation. Notice that this change makes the simulator abort with the same probability that an honest player would abort in Step 3.

We now argue that when there is no abort, the output of Π^{MS} is exponentially close to that of $\mathfrak{I}^{MS}$. Notice that picking the disjoint $S_2, ..., S_k \subseteq [\ell]$ uniformly at random is equivalent to first picking $\{S_i\}_{i\in I_{\mathcal{A}}}$ from $[\ell]$, and then picking $\{S_i\}_{i\notin\mathcal{A}}$ from the remaining $[\ell] \setminus \left(\bigcup_{i\in\mathcal{A}} S_i\right)$ elements. From this perspective, if the honest players do not abort in Step 3, then Lemma I.1 implies that the state created by player 1 in the other positions $[\ell] \setminus \left(\bigcup_{i\notin\mathcal{A}} S_i\right)$ is $O(2^{\varepsilon^2(k-|I_{\mathcal{A}}|)n})$-close to the the subspace $\operatorname{span}\left((P_\pi |\mathsf{T}\rangle^{\otimes tn-j} |\mathsf{T}^\perp\rangle^{\otimes j}) : 0 \leqslant j \leqslant \varepsilon tn, \pi \in S_{tn}\right)$. If we choose $\varepsilon \leqslant \frac{1}{2}\left(1 - \sqrt{3/7}\right)$, by Lemma 2.7 and the union bound, the output of the distillation procedure is $O\left(t\varepsilon\right)^{n^c}$-close to $|\mathsf{T}\rangle^{\otimes t}$. In this case, the output of Π^{MS} will be $\mathsf{negl}\left(n\right)$ close to encodings of $|T\rangle^{\otimes t}$, which is the output of $\mathfrak{I}^{MS}$ in the no-abort case. □