Asynchronous Multi-Party Quantum Computation

Vipul Goyal
Carnegie Mellon University, Pittsburgh, PA, USA
NTT Research, Sunnyvale, CA, USA

Chen-Da Liu-Zhang
NTT Research, Sunnyvale, CA, USA

Justin Raizes
Carnegie Mellon University, Pittsburgh, PA, USA

João Ribeiro
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Multi-party quantum computation (MPQC) allows a set of parties to securely compute a quantum circuit over private quantum data. Current MPQC protocols rely on the fact that the network is synchronous, i.e., messages sent are guaranteed to be delivered within a known fixed delay upper bound, and unfortunately completely break down even when only a single message arrives late.

Motivated by real-world networks, the seminal work of Ben-Or, Canetti and Goldreich (STOC'93) initiated the study of multi-party computation for classical circuits over asynchronous networks, where the network delay can be arbitrary. In this work, we begin the study of asynchronous multi-party quantum computation (AMPQC) protocols, where the circuit to compute is quantum.

Our results completely characterize the optimal achievable corruption threshold: we present an $n$-party AMPQC protocol secure up to $t < n/4$ corruptions, and an impossibility result when $t \geq n/4$ parties are corrupted. Remarkably, this characterization differs from the analogous classical setting, where the optimal corruption threshold is $t < n/3$.

2012 ACM Subject Classification Hardware → Quantum communication and cryptography

Keywords and phrases Quantum Cryptography, Multiparty Computation, Asynchronous

Digital Object Identifier 10.4230/LIPIcs.ITCS.2023.62

Related Version Full Version: https://eprint.iacr.org/2022/1583

Funding Research supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.

Chen-Da Liu-Zhang: Work partially done while the author was at Carnegie Mellon University.

1 Introduction

Secure multi-party computation (MPC) allows a set of parties to compute a function of their private inputs, in such a way that the parties’ inputs remain as secret as possible, even in the presence of an adversary corrupting a subset of the parties.

The problem of MPC has been studied mostly in classical setting, where the function to evaluate as well as the adversary are classical [42, 25, 9, 14, 38]. However, with the recent advances on quantum computing technology, it has become increasingly relevant to consider quantum functionalities and quantum adversaries. This motivated an important line of works on multi-party quantum computation (MPQC) protocols [21, 8, 41, 24, 23, 6, 4].

Current MPQC protocols operate in the so-called synchronous network model, where parties have access to synchronized clocks and there is an upper bound on the network communication delay $\Delta$. Although this model is theoretically interesting, it fails to capture...
real-world networks such as the Internet, which is inherently asynchronous. In fact, assuming a synchronous network is arguably worse in the quantum world, given the difficulties in maintaining quantum states coherently. This has catastrophic consequences, since the security of synchronous protocols is often completely compromised as soon as even one message is delayed by more than $\Delta$ time.

In contrast, protocols in the asynchronous network model do not rely on any timing assumptions, and messages sent can be arbitrarily (and adversarially) delayed. While asynchronous MPC protocols in the classical setting have been known since a few decades, no protocol has been proposed in the quantum setting. An inherent difficulty in such protocols is that one cannot distinguish between a dishonest party not sending a message, or an honest party that sent a message that was delayed by the adversary. As a result, parties have to make progress in the protocol after receiving messages from $n - t$ parties. This also implies that in this setting it is impossible to consider the inputs of all honest parties – the inputs of up to $t$ (potentially honest) parties may be ignored. Asynchronous protocols impose even further restrictions in the quantum setting. In particular, since states cannot be duplicated, the no-cloning theorem rules out quantum broadcast, which is a crucial tool for enabling classical asynchronous protocols.

In the classical setting, the foundational works of Ben-Or, Canetti and Goldreich [7] and Ben-Or, Kelmer and Rabin [10] showed that the optimal achievable corruption tolerance in the asynchronous model is $t < n/3$. In this work, we initiate the study of the quantum counterpart, namely asynchronous multi-party quantum computation (AMPQC) protocols:

Is it possible to achieve AMPQC? If so, what is the optimal achievable corruption threshold?

We completely resolve this question by providing the first AMPQC protocol secure up to $t < n/4$ corruptions, and showing a lower bound that tolerating $t \geq n/4$ corruptions is impossible.

**Theorem 1 (AMPQC Feasibility).** There exists an information-theoretic asynchronous multiparty quantum computation protocol for $n$ parties which is secure against up to any $t < \frac{n}{4}$ corruptions.

**Theorem 2 (AMPQC Impossibility).** No asynchronous multiparty quantum computation protocol exists for $n$ parties which tolerates any corruption threshold $t \geq \frac{n}{4}$.

### 1.1 Related Work

**Classical Asynchronous MPC**

The seminal works of Ben-Or, Kelmen, and Rabin [10], and Ben-Or, Canetti, and Goldreich [7] showed that the optimal achievable corruption tolerance for asynchronous classical MPC is $t < n/3$ even with setup, in both the computational and information-theoretic settings, and $t < n/4$ when requiring perfect security.

Since then, a huge amount of work has been devoted to improving the communication complexity. In the information-theoretic setting with optimal resilience, Patra, Choudury, and Pandu Rangan [35] achieved $O(n^5 \kappa)$ bits per multiplication, where $\kappa$ is the security parameter, and recently Choudhury [16] further improved this result to $O(n^4 \kappa)$ bits per multiplication. Going to the sub-optimal resilience $t < n/4$, several works achieve linear communication [40, 37, 17, 36].

In the cryptographic setting, most protocols make use of some form of homomorphic encryption. The works by Hirt, Nielsen, and Przydatek [28, 29] make use of additive homomorphic threshold encryption, with the protocol in [29] communicating $O(n^2 \kappa)$ bits per
multiplication, and the work by Chopard, Hirt, and Liu-Zhang [15] achieves adaptive security with the same communication. The work by Choudhury and Patra [18] achieves $O(n\kappa)$ per multiplication using somewhat-homomorphic encryption, and several other works [20, 31, 12] make use of fully-homomorphic encryption to achieve communication complexity independent of the circuit size.

Multi-Party Quantum Computation

All known MPQC protocols assume a synchronous network and have a negligible error probability. Crépeau, Gottesman, and Smith [21] introduced the first MPQC protocol with guaranteed output delivery up to $t < n/6$ corruptions. This was improved by Ben-Or, Crépeau, Gottesman, Hassidim, and Smith [8], who achieved the optimal resilience $t < n/2$. In the dishonest majority setting, where up to $n-1$ parties may be corrupted, Dulek, Grilo, Jeffery, Majenz, and Schaffner [23] gave the first MPQC protocol achieving security with abort, building upon the work by Dupuis, Nielsen and Salvail that achieved two-party secure computation. Very recently, in a followup work, Alon, Chung, Chung, Huang, Lee, and Shen [4] designed a protocol with identifiable abort. Another recent work by Bartusek, Coladangelo, Khurana, and Ma [6] achieves a constant round MPQC protocol in the dishonest majority setting.

Our work achieves guaranteed output delivery up to $t < n/4$ corruptions under an asynchronous network and incurs negligible error. It is left as an open question whether one can achieve MPQC with perfect security (meaning 0 error probability), even for synchronous networks.

Post-Quantum Computation

A line of works has focused on the problem of post-quantum computation which considers classical computation, but an adversary with quantum capabilities.

As noted in [1], many of the results in the classical setting [9, 14, 13, 30] can be proven post-quantum secure, provided they are instantiated using primitives that are plausibly quantum secure. Damgård and Lunemann [22] introduced a two-party coin-flipping protocol, and Lunemann and Nielsen [32] and Hallgren, Smith and Song [26] introduced general two-party computation protocols secure against quantum adversaries. Bitansky and Shmueli [11] gave a constant-round two-party coin-flipping protocol, with full simulation of one party. Finally, very recently, Agarwal, Bartusek, Goyal, Khurana and Malavolta [1] introduced a constant-round post-quantum multi-party computation protocol in the plain model.

2 Technical Overview

We give an overview of the main techniques used in the paper.

2.1 Feasibility Result

We provide a protocol secure up to $t < n/4$ corruptions under an asynchronous network. Our starting point is the work by Ben-Or, Crépeau, Gottesman, Hassidim, and Smith [8], which achieves optimal resilience up to $t < n/2$ when the network is synchronous.

Their protocol follows the traditional sharing-based paradigm for multi-party computation [9]. Parties distribute their private inputs with a so-called verifiable quantum secret sharing scheme (VQSS). Parties then evaluate the circuit in a gate-by-gate fashion on the encoded inputs. In the end, parties end up with encodings of the outputs of the circuit,
Asynchronous Multi-Party Quantum Computation

which can each be jointly decoded towards the corresponding party. Technically, the main challenge comes from the design of a VQSS which is in some sense compatible with quantum operations. For this, the authors elegantly design a VQSS which relies on several building blocks, including a special quantum authentication scheme and a so-called weak quantum secret-sharing scheme (WQSS), which is similar to VQSS, except that a dishonest dealer can choose not to reconstruct the shared secret (i.e. reconstruct ⊥). The design allows to push the complexity of performing quantum operations over encoded data to performing classical operations on the authentication keys.

Roughly speaking, the VQSS scheme is composed of three parts: First, the dealer shares $2\kappa + 2$ quantum $|0\rangle$-states among the parties, and each share is then re-distributed using a WQSS scheme. Second, parties jointly run a checking phase to verify that they indeed hold a sharing of $|0\rangle$ (except with error negligible in $\kappa$). This is accomplished with a so-called zero-purity test, which can be run using a classical trusted third party (TTP). The zero-purity test requires each party to do some local computation before measuring $2\kappa$ of the shared states and sending the measurement results to the TTP, which combines them. This consumes $2\kappa$ of the shared states. Finally, with the remaining two sharings of $|0\rangle$, the parties jointly create sharings of an EPR pair. The first half of the EPR pair is decoded to the dealer, who can then distribute the secret to the parties using quantum teleportation.

Their scheme relies on a synchronous network, where every message sent by an honest party is delivered within a known delay upper bound, and unfortunately it completely breaks down as soon as even one message gets delayed. In fact, an honest dealer could appear corrupted because his messages did not arrive in time. The recipients will then accuse him during the checking phase, resulting in the sharing protocol being unsuccessful.

Challenges in the Asynchronous Setting

In order to make this work in the asynchronous setting, the main obstacle is to design a mechanism that allows parties to learn when they jointly have enough information to uniquely determine the shared secret. When the network is synchronous, this is straightforward, given that parties proceed synchronously and the secret is guaranteed to be shared after a certain number of rounds. In asynchrony, some parties might have gotten a lot of messages while others might still be waiting. Note that if a party $P_i$ didn’t receive a message from another party $P_j$, $P_i$ cannot ask for a missing message, since the response can also be delayed.

In the classical setting, asynchronous VSS is typically solved by reaching agreement on a core-set of parties of size at least $n - t$ who received correct shares. The reasoning is that if $n - 2t > t$, then the core-set contains at least $t + 1$ honest parties, whose shares should uniquely define the secret. The parties in the core-set will be the only ones that contribute their shares in the reconstruction step. One usually also requires a way to tell during the reconstruction phase whether a received share is correct (this is usually achieved using authentication tools such as information checking or signatures, see, e.g., [10, 34]), which allows parties to correctly reconstruct the secret with high resilience. Three challenges arise when trying to achieve this, which we discuss below.

Using a Classical TTP. First, in order to agree on a core-set, we make use of an asynchronous trusted third party (TTP) that can perform classical computations. Perhaps surprisingly, this does not follow from standard classical asynchronous MPC in a black-box way. This is

---

1 In fact, it is more complicated than that: In many synchronous protocols, if $P_i$ doesn’t receive a message from $P_j$, $P_j$ is deemed corrupted and the protocol can for example reveal $P_j$’s secret state.
because asynchronous MPC protocols have a very concrete interaction pattern where the TTP waits for the values of any $n - t$ parties (which the adversary can choose by scheduling messages), and outputs a function of these values. However, the interaction that we need with the TTP is more general: in particular, the TTP can wait for the values of designated parties to arrive before computing the function. For example, in the broadcast functionality a designated party sends a message to the TTP, and then the TTP forwards that message to all other parties.

In order to deal with more general functionalities, we first formalize a generalized secure function evaluation protocol, which not only takes into account the function $f$ to be evaluated, but also a monotone predicate $Q : P \rightarrow \{0, 1\}$ (i.e., if $T \subseteq T'$ and $Q(T) = 1$, then $Q(T') = 1$) that indicates which sets of parties’ inputs may be included into the computation. Using generalized secure function evaluation protocols, one can realize the classical TTP using standard techniques, where each evaluated function takes into account the internal state of the TTP, which is jointly maintained by the parties.

We then show how to modify current existing classical asynchronous protocols for secure function evaluation to achieve generalized secure function evaluation. Current information-theoretic protocols for secure function evaluation follow the traditional sharing-based paradigm, and distribute the inputs as follows: Parties use an asynchronous verifiable secret sharing (AVSS) scheme. Then, since the network is asynchronous, some sharings terminate earlier than others, and therefore parties need to agree on when to proceed to the computation phase. For that, parties run a core-set agreement protocol [10, 7] to agree on a core-set of parties of size at least $n - t$ whose inputs are taken into the computation (all other parties’ inputs are ignored). In order to take the inputs into account according to a predicate $Q$, one can proceed as follows: run $n$ Byzantine Agreement (BA) protocols $BA_1, \ldots, BA_n$, one for each party. Every time the AVSS from party $P_j$ terminates, $P_i$ inputs 1 to $BA_j$. Every time $BA_j$ outputs 1, it adds $P_j$ to his local set $T_i$. Party $P_i$ then waits until the set of parties $T_i$ satisfies $Q(T_i) = 1$. If so, $P_i$ inputs 0 to all remaining BAs, and waits for all BAs to terminate before proceeding to the computation phase. Due to agreement of BA, all honest parties agree on the same set of parties $CoreSet$. Moreover, all honest parties eventually receive all the inputs from parties in $CoreSet$, due to properties of the AVSS. Finally, since $Q$ is monotone, it follows that the final set of inputs taken into account for the computation satisfies $Q$ (as it contains at least the set $T_i$). For more details, see the full version.

**Reconstruction and Corruption Thresholds.** Second, in contrast to the classical setting, a quantum secret sharing scheme cannot have reconstruction threshold $t + 1$, because the no-cloning theorem enforces that the reconstruction threshold must always be at least $\lceil n/2 \rceil + 1$ [19]. Due to the asynchronous nature of the network, the reconstructor must be able to perform the reconstruction process with shares from only $n - t$ parties, since the protocol must succeed even if the $t$ adversarial parties refuse to participate. However, in order to uniquely define the secret, $\lceil n/2 \rceil + 1$ of those shares must be from honest parties because of the reconstruction threshold. Combining these observations with the fact that $t$ of the provided shares can be from corrupted parties imposes the requirement that $n - 2t \geq \lceil n/2 \rceil + 1$, or $t \leq n/4$. Note that in the classical setting, setting the reconstruction threshold to $\lceil n/3 \rceil + 1$ allows $t \leq n/3$. We later expand this intuition into a proof of impossibility for $t \geq n/4$. 

---

2 In a traditional asynchronous protocol, $Q$ evaluates to 1 for any set of size at least $n - t$. 
Robust Reconstruction. Finally, even for $t < n/4$ and assuming a classical TTP, it is not clear how to robustly reconstruct a secret (even in the strictly weaker setting of WQSS).

To see this, let us say that one follows the classical approach of defining a core-set of $n - t$ parties who received correct shares and should contribute during the reconstruction phase. Among these, given that $t$ parties may be corrupted, parties cannot expect to receive all and must reconstruct already from $n - 2t$ shares. However, the $n - 2t$ received shares may contain $t$ corrupted shares. Given that one needs at least $\lfloor n/2 \rfloor + 1$ shares to reconstruct the secret, one can only safely reconstruct if $t < \frac{n}{6}$. In this case, there are $n - 2t - |\text{errors}| = n - 3t > n/2$ honest shares, which is enough to uniquely define the secret. However, a higher corruption threshold might result in the honest shares not uniquely defining the secret, in which case the corrupted parties might be able to force a different reconstruction value. Therefore achieving the optimal threshold of $t < \frac{n}{4}$ requires additional ideas.

In the classical setting, this issue is addressed by letting each share be signed by every party during the sharing phase. This ensures that during the reconstruction the adversary cannot send a corrupted share (with correct signatures) corresponding to a different secret. Unfortunately, there is no easy way to achieve this in the quantum setting (digitally signing quantum data is even impossible! [5, 3]). To overcome this barrier we introduce a novel primitive called asynchronous weak quantum secret-sharing scheme (AWQSS) with weak termination, which does not guarantee that the reconstruction procedure outputs a value (even when the sharing was successful) in the dishonest-dealer case.\(^3\) We then show how this weaker primitive is enough to achieve AVQSS.

The starting point for the AWQSS with weak termination is the synchronous protocol from Ben-Or et al. [8]. The dealer verifiably distributes authenticated shares of two $|0\rangle$ states, where the classical TTP holds the classical authentication key. With the help of the TTP, the parties test the two sets of shares to ensure they are actually shares of $|0\rangle$ states. Once assured, the parties entangle the two shared qubits to create a shared EPR pair and send half to the dealer. Finally, the dealer uses the EPR half it receives to teleport its state. Since the protocol is required to progress with only $n - t$ active participants, the $|0\rangle$ test must occur as soon as any $n - t$ parties have provided their test measurements.

Contrary to the synchronous WQSS case, the requirement that progression is necessary with only $n - t$ parties opens up problems in aligning the cases when the dealer is honest and when it is corrupt. Specifically, it becomes possible for $t$ honest parties to receive inconsistent shares (or not receive them at all) without disrupting the sharing protocol. Then, depending on the behavior of the corrupted parties during reconstruction, the reconstructor may only receive $n - 2t$ shares. Since the reconstructor cannot tell whether the remaining shares are simply delayed on the network or withheld by corrupt parties, it must decide whether to attempt to reconstruct the secret or output $\perp$ based only on these $n - 2t > \lfloor n/2 \rfloor + 1$ shares. However, even though $n - 2t$ is at least the threshold of $\lfloor n/2 \rfloor + 1$ for $t < n/4$, it is never safe to attempt to reconstruct the secret, even if the received shares are consistent. This is because if the dealer is corrupt, then they know the authentication keys and so the other corrupt parties may provide up to $t$ arbitrary shares in the reconstruction process, potentially changing the reconstructed value. To make matters worse, the same situation can occur with an honest dealer, but reconstruction must always succeed in this case!

\(^3\) In ordinary weak secret-sharing, as is used in prior work, the reconstructor is allowed to output $\perp$, but must terminate at some point. With weak termination, the reconstructor may even fail to output $\perp$ when the dealer is dishonest.
Our insight here is to build a late checking mechanism into the protocol which allows parties outside of the core-set to contribute. These parties did not participate in the initial share-checking, but will still hold valid shares in the case of an honest dealer. This is in contrast to current classical asynchronous protocol techniques, where only parties inside the core-set contribute their shares in the reconstruction step.

Honest parties who receive their shares after the $|0\rangle$ test has already occurred can send their portion of the test measurements to the TTP and receive back the result of performing the $|0\rangle$ test on their measurement and $n-t-1$ of the original $n-t$ test measurements used in the main $|0\rangle$ test. This TTP behavior makes use of the expanded classical TTP functionality discussed above in order to allow a single party to interact with the TTP. With an honest dealer, the $t$ honest parties who are delayed will have their shares confirmed by the TTP after the main check has already occurred. This allows $n-t$ shares to be provided to the reconstructor, which is sufficient to complete reconstruction safely. It is important to note that late checking does not change the case for a corrupted dealer, since the delayed honest parties might never receive shares in the first place.

AVQSS avoids the weak termination problem by creating a two-level sharing of $|0\rangle$. Creating a two-level sharings of $|0\rangle$ intuitively means that the quantum state $|0\rangle$ is shared using an AWQSS scheme, and each level-1 share is again shared using an AWQSS scheme. The sharing terminates once at least $n-t$ parties hold correct shares, as checked by another $|0\rangle$ test. These $n-t$ parties uniquely define the secret. Moreover, during the reconstruction, corrupted parties cannot send corrupted level-1 shares, since each of these shares are distributed among all honest parties. They can, however, withhold their shares. But since the $n-t$ parties contain at least $n-2t \geq \lfloor n/2 \rfloor + 1$ honest parties, these are enough to reconstruct the secret.

### 2.2 Impossibility Result

As we saw above, we require the corruption threshold to be $t < n/4$ in our protocol. Interestingly, we show that this corruption threshold is optimal. This is in contrast to the classical setting, where the optimal threshold for asynchronous computation is $t < n/3$ [10].

Formally, we prove that AVQSS is impossible for $t \geq n/4$. The ideas in our proof generalize naturally to a secure function evaluation protocol for all-to-all AVQSS, where all the parties end up with shares from each of the $n-t$ parties in the core-set. We provide a high level idea of the proof below for $n=4$ and $t=1$. By standard arguments, this implies an impossibility for the general case $t \geq n/4$.

Consider the existence of an AVQSS protocol with four parties $D, P_2, P_3, P_4$, where $D$ also acts as a recipient, and secure up to $t=1$ corruption. We show that this implies an “approximate” quantum erasure-correcting code (QECC) of length 4 and approximately correcting 2 erasures, in the sense that the decoded quantum state is close in trace distance to the true input state. We prove that it is impossible to construct such codes. The idea is that there can be one corrupted party, and, because the protocol succeeds under an asynchronous network, it must succeed even when a potentially honest party is locked out of the protocol. Formally showing this intuition requires carefully designing a scenario-based argument.

A bit more formally, consider a secret $x$, and further consider without loss of generality the set $\{P_3, P_4\}$ (the argument holds for any set of size 2). We will show that the internal state of $\{P_3, P_4\}$ fully determines $x$ (up to a small error).

---

4 The test is defined for all parties, although it does not require all of them to participate.
Consider a first scenario where all parties are honest in the execution of the sharing phase, but \( P_2 \)'s messages are delayed. Since the protocol is secure even when \( P_2 \) crashes from the start, all parties successfully terminate holding a share. In this case, since the dealer is honest, the reconstructed value is \( x \).

In the second scenario, \( P_2 \) does not receive any information from a corrupted dealer \( D \) in addition to having its messages delayed. The dealer \( D \) otherwise behaves as in an honest execution with input \( x \) with respect to parties \( \{P_3, P_4\} \) (by internally emulating \( P_2 \)). In this case, the view of \( \{P_3, P_4\} \) is exactly the same as in the first scenario. Furthermore, note that there is an adversarial strategy so that the reconstructed value is \( x \): simply let \( D \) also participate in the reconstruction protocol honestly while delaying all messages from \( P_2 \), as this is the same execution as in the first scenario. In turn, this means that the committed value is \( x \), and, regardless of the adversarial strategy, the reconstructed value must be \( x \).

However, since \( P_2 \) did not receive any message from \( D \), his internal state cannot provide any information that is unknown to \( \{P_3, P_4\} \). Since AQVSS requires the views of the honest parties to define the secret, the internal state of \( \{P_3, P_4\} \) must fully determine the secret \( x \).

This implies the existence of an approximate QECC with length 4 that is resilient to two erasures, since the secret can be recovered from any two shares.

3 Preliminaries

3.1 Notation

We denote the security parameter by \( \kappa \) and write \( \text{neg}(\kappa) \) for any function that is negligible in \( \kappa \), i.e., \( \text{neg}(\kappa) \) decays faster than \( \kappa^{-c} \) for any constant \( c > 0 \) as \( \kappa \) grows. We also write \( \text{poly}(\kappa) \) for a function that grows polynomially with \( \kappa \). We write \( k \leftarrow K \) to mean that \( k \) is sampled uniformly at random from the set \( K \). The finite field of order \( q \) is denoted by \( \mathbb{F}_q \).

The conjugate transpose of a matrix \( U \) is denoted by \( U^\dagger \).

3.2 Concepts from Quantum Computation

We assume familiarity with basic concepts from quantum computation, such as pure and mixed states, density matrices, Clifford and Toffoli gates, entanglement, measurements, and quantum teleportation. We refer the reader to the book of Nielsen and Chuang [33] for an overview of these concepts. We denote quantum registers and gates by uppercase roman letters. The distinction will be clear from context. Density matrices are denoted by lowercase greek letters such as \( \rho \) and \( \sigma \), and we sometimes write \( \rho_M \) for the state associated with register \( M \). Furthermore, we write \( U_M \) for a unitary transformation \( U \) to denote that \( U \) is applied to the contents of register \( M \).

Trace Distance

We make use of the notion of trace distance between states.

▶ Definition 3 (Trace distance). The trace distance between two mixed states with associated density matrices \( \rho \) and \( \sigma \), denoted by \( D(\rho, \sigma) \), is given by

\[
D(\rho, \sigma) = \frac{1}{2} \| \rho - \sigma \|_1,
\]

where \( \| \rho \|_1 = \text{Tr}[\sqrt{\rho^\dagger \rho}] \) is the trace norm.
The trace distance is a distance and has the following useful interpretation: If $D(\rho, \sigma) \leq \varepsilon$, then any POVM applied to states with density matrices $\rho$ and $\sigma$ yields classical measurement outcome distributions, say $(p_1, \ldots, p_m)$ and $(q_1, \ldots, q_m)$, which are $\varepsilon$-close in statistical distance, i.e., $\frac{1}{2} \sum_{i=1}^{m} |p_i - q_i| \leq \varepsilon$.

**Generalized Quantum Gates**

We work with basis states indexed by elements of a finite field $\mathbb{F}_p$ and apply several quantum operations to such states. We describe them next. We consider the generalized Pauli gates $X$ and $Z$ over $\mathbb{F}_p$, which act as $X|\alpha\rangle = |\alpha + 1\rangle$, where the sum is over $\mathbb{F}_p$, and $Z|\alpha\rangle = \omega_p^\alpha |\alpha\rangle$, where $\omega_p = e^{2\pi i/p}$. Moreover, we will use the controlled SUM gate (generalizing the CNOT gate) over $\mathbb{F}_p$ acting as $\text{SUM}|\alpha, \beta\rangle = |\alpha, \alpha + \beta\rangle$, and the $\gamma$-Fourier gate $F_\gamma$ over $\mathbb{F}_p$ acting as $F_\gamma|\alpha\rangle = p^{-1/2} \sum_{\beta \in \mathbb{F}_p} \omega_p^{\gamma \alpha \beta} |\beta\rangle$.

When $\gamma = 1$ we simply write $F = F_1$.

**3.3 Quantum Secret Sharing**

Quantum secret sharing [27] allows a dealer to share a secret quantum state so that authorized subsets can recover the secret, but unauthorized sets cannot gain information about the secret. In our schemes, we use an extension of Shamir’s secret sharing scheme to the quantum setting, first described by Cleve, Gottesman, and Lo [19]. Note that the no-cloning theorem implies that we must always have $t > n/2$ in a QTSS scheme [19].

**3.4 Quantum Authentication Schemes**

Quantum authentication schemes were first introduced in [5]. Intuitively, a quantum authentication scheme encodes a quantum state with the help of a classical key so that operations performed on the authenticated state by an adversary who does not know the key can be detected.

**Polynomial-Based Authentication Scheme**

We will be using the efficient polynomial-based authentication scheme from [8] and based on [2]. This scheme has several properties which will be quite useful. First, the authentications keys $(k, x)$ are classical, which allows them to be managed by the classical TTP. This advantage is especially important because their scheme allows the application of Clifford gates and measurements to authenticated states where the keys are held by the TTP and the quantum registers are held by other parties. In the case of multi-qubit gates, states authenticated under the same $k$ (but potentially different $x$) can be operated on jointly. Second, the scheme remains secure in a setting where the adversary has access to several states authenticated with the same $k$ but independently sampled $x$’s. This will later allow parties to authenticate multiple states which can all be operated on together.

**4 Model of Multiparty Computation**

We consider a set of $n$ parties $\mathcal{P} = \{P_1, \ldots, P_n\}$. We extend the standard classical asynchronous model for multiparty computation to the quantum setting, where parties have quantum states as inputs and wish to apply a quantum circuit to their states.
We consider quantum circuits which use only Clifford and Toffoli gates, since these form a universal quantum gate set [39].

4.1 Communication and Adversarial Models

Parties have access to point-to-point private classical and quantum channels. For simplicity, we consider a static computationally unbounded adversary who is allowed to corrupt any $t$ parties at the start of the protocol, who may deviate arbitrarily from the protocol. But we believe our protocols should also achieve adaptive security.

Our network model is asynchronous. This means that there may be an arbitrary (finite) delay between the time a message is sent and the time it is delivered. The adversary controls the scheduling of the messages, subject only to the constraint that every message sent must eventually be delivered. Since we assume private channels, the adversary has no information about the contents of each message besides the identity of the sender and receiver.

4.2 Multi-Party Asynchronous Quantum Computation

We adapt the security definition to our setting, following the works on asynchronous classical MPC by Ben-Or, Canetti, and Goldreich [7] and synchronous MPQC by Dulek, Grilo, Jeffery, Majenz, and Schaffner [23]. The security definition introduces a real and an ideal world, and intuitively guarantees that any attack that happens in the real-world can be efficiently reproduced in the ideal-world. In the real world, the protocol is run between the honest parties and the adversary. In the ideal world, the adversary interacts with an ideal functionality which takes in the inputs of every party, then outputs the result of the computation. During this interaction, it may choose up to $t$ parties to exclude from the computation. These parties may be honest.

4.3 Classical Trusted Third Party

We make use of a classical trusted third party in our protocol which is stateful and reactive. Upon receiving any message from any party, it parses the message type and responds according to the message contents accordingly. For ease of exposition, we present the reactions of the TTP modularly alongside the corresponding sub-protocols. When the TTP would be initialized with the same information at multiple steps in a protocol, it should be understood that it is initialized once with this information and saves it as internal state.

Realizing a classical asynchronous TTP does not follow directly from protocols for secure function evaluation, since the evaluated function takes into account the inputs of any $n-t$ parties. In some cases, the interaction with the TTP can include instructions that allow the TTP to wait for inputs from designated parties. In the full version, we show how to achieve such a classical asynchronous TTP.

5 Protocols

5.1 Verified Quantum State Authentication

The verified authentication of zeros protocol allows each honest party to receive an authenticated $|0\rangle$ state from some dealer and to be certain that all other honest parties will eventually receive a $|0\rangle$ state authenticated under the same key. These $|0\rangle$ states can then be transformed into an arbitrary authenticated state using quantum teleportation. Later, this will act as a way to allow honest parties to prove to the classical TTP that they provided measurements of some state received from the dealer.
At a high level, the dealer prepares and authenticates many $|0\rangle$ states. They send the classical authentication keys to the TTP, then send each party some of the $|0\rangle$ states. With the help of the TTP, each party tests the states it received using the zero purity testing protocol from [8]. If the test passes, the party keeps one of the $|0\rangle$ states and forwards the rest to the other parties. If it fails, the party waits to receive forwarded states, which it then tests again.

The following lemma states some important properties of the proposed authentication protocol. The full construction and proof can be found in the full version of the paper.

**Lemma 4.** With probability at least $1 - \text{neg}(\kappa)$, the following properties hold in an execution of the Verified Authentication of Zeros protocol:

- Each honest party which outputs holds $r$ states with trace distance $\text{neg}(\kappa)$ to $|0\rangle$ states authenticated under the keys $(k, \vec{x})$ held by the classical TTP.
- If any honest party outputs, then all honest parties do so.
- If the dealer is honest then all honest parties output.

Moreover, these properties continue to hold even when composed in parallel with another Authentication of Zeros protocol which shares the same dealer key $k$ (and independent $\vec{x}$).

### 5.2 Asynchronous Weak Quantum Secret Sharing with Weak Termination

The next building block for asynchronous verifiable quantum secret sharing is asynchronous weak quantum secret sharing (AWQSS), which is described by a pair of protocols $(\text{Share}, \text{Reconstruct})$. In the first protocol, $\text{Share}$, a designated party called the dealer $D$ distributes a private input state $s$ among the set of parties. In the second protocol, $\text{Reconstruct}$, the parties jointly participate and a designated receiver $R$ obtains the secret. In weak secret sharing, $R$ is also allowed to output $\perp$ when the dealer is dishonest. The protocol we describe uses a trusted party which performs classical computation.

**Definition 5 (Asynchronous Weak Quantum Secret Sharing with Weak Termination).** Consider a pair of protocols $(\text{Share}, \text{Reconstruct})$ for $n$ parties, where a designated party $D$, called the dealer, has a private input quantum state $s$ for $\text{Share}$, and each honest party that completes $\text{Share}$ subsequently invokes $\text{Reconstruct}$ with its local output of $\text{Share}$, and a designated reconstructor $R$ outputs a quantum state upon terminating. We say that $(\text{Share}, \text{Reconstruct})$ is a $(t, \varepsilon)$-secure asynchronous weak quantum secret sharing scheme if the following holds with probability at least $1 - \varepsilon$ whenever up to $t$ parties are corrupted:

- **Termination:**
  - If $D$ is honest, then every party eventually terminates $\text{Share}$. Moreover, $R$ outputs a state at the end of $\text{Reconstruct}$.
  - If some honest party terminates $\text{Share}$, then all honest parties eventually terminate $\text{Share}$.

- **Privacy:** If $D$ is honest, the view of the adversary is independent of $s$.

- **Correctness:** If $D$ is honest, then if $R$ outputs a state at the end of $\text{Reconstruct}$, the state is $s$.

- **Commitment:** Even if $D$ is corrupted, if all honest parties terminate $\text{Share}$, their joint views defines a unique state $s'$, such that if $R$ outputs a state after $\text{Reconstruct}$, the state has $\text{neg}(\kappa)$ trace distance to either $s'$ or $\perp$.

Note that if $D$ is corrupted, an honest reconstructor $R$ is not guaranteed to obtain an output from the protocol $\text{Reconstruct}$. We emphasize that this occurs only at the level of weak secret sharing and does not propagate to verifiable secret sharing.

We first define a trusted classical party that we use in the protocol AWQSS.
## Functionality AWQSS Classical TTP

**Initialization step.** Let \( s = \text{poly}(\kappa) \). The TTP receives keys \((k, \vec{x})\) from the dealer \(D\). It sets \( S = \emptyset \) and \( \text{result} = \bot \).

**Sharing Execution** Upon receiving \((\text{ShareCheck}, m)\) from \(P_i\), where \(m\) is a set of measurements and \(P_i\) has not sent a ShareCheck message before, the TTP does the following:

1. Verify the authentication code on \(m\) using \((k, \vec{x})\). If it fails, discard this message. Otherwise, continue:
2. if \(|S| < n - t\) then
3. \( S \leftarrow S \cup \{i\} \)
4. If \(|S| = n - t\), perform the zero purity test using \(m\) and set store the result in \(\text{result}\). Then send \((\text{ShareCheck}, \text{result})\) to each \(P_j\) for \(j \in S\).
5. else
6. Perform the purity test \(2s\) times using \(m\).
7. Send \((\text{ShareCheck}, \text{acc})\) to \(P_i\) if this test accepts and \(\text{result} = \text{acc}\). Otherwise send \((\text{ShareCheck}, \text{rej})\) to \(P_i\).

**Reconstruction Execution** The TTP sends \((\text{Reconstruct}, k, \vec{x})\) to the reconstructor \(R\).

## Protocol AWQSS

We now describe the protocol from the point of view of party \(P_i\), then the final sharing from the point of view of the dealer \(D\).

**Initialization step.** Let \( s = \text{poly}(\kappa) \). The dealer \(D\) chooses a random key \(k_D\). It generates \(2s + 2|\emptyset\rangle\) states for \(s = \text{poly}(\kappa)\) and shares them using quantum Shamir secret sharing with threshold \(\lfloor \frac{n}{2} \rfloor + 1\) and \(n\) shares. For each encoded \(|\emptyset\rangle\) state, send the \(i\)-th share to \(P_i\) using the key \(k_D\) and freshly random \(x\) in the Verified Authentication protocol. Let \(\vec{x}\) be the vector containing each \(x\) which is used. \(D\) sends the keys \((k, \vec{x})\) to the TTP.

**Sharing Execution.**

1. Let \(R_1, \ldots, R_{2s+2}\) be the registers containing the (authenticated) shares which \(P_i\) received from \(D\).
2. Perform the zero purity test measurements transversally, storing the results in \(m\).
3. \(P_i\) sends \((\text{ShareCheck}, m)\) to the TTP. Implicitly, the TTP finishes the zero purity test.
4. \(P_i\) waits to receive \((\text{ShareCheck}, \text{result})\) from the TTP.
5. If \(\text{result}\) is \(\text{acc}\), then \(P_i\) generates an EPR pair share using the two remaining \(|\emptyset\rangle\) shares in \(R_0, R_1\). Then send the share in \(R_0\) to \(D\). Otherwise abort

**Teleportation**

1. \(D\) waits until it receives \(\frac{n}{2}\) correctly authenticated shares from the parties, then uses them to reconstruct the EPR pair half (this is possible since it knows the keys and can update them according to the EPR construction, which is deterministic).
2. \(D\) uses the reconstructed EPR pair half to teleport the contents of \(R_D\), sending the measurement results to the TTP.
3. The TTP informs all parties that the sharing is over.
Reconstruct
1: All parties send their shares to the reconstructor $R$.
2: $R$ waits to receive the classical keys classical keys $(k, \vec{x})$ from the TTP.
3: $R$ checks the authentication on each received share and discards any which do not authenticate correctly.
4: $R$ waits until $n - t$ correctly authenticated shares are received.
5: $R$ removes the authentication on these shares.
6: these shares all lie on the same polynomial superposition, $R$ reconstructs and outputs the state using an arbitrary $\lfloor \frac{n}{t} \rfloor + 1$ of them. Otherwise, $R$ outputs $\perp$.

Lemma 6. Protocol AWQSS is a $(t, \text{neg}(\kappa))$-secure asynchronous weak quantum secret sharing scheme for all $t < \frac{n}{4}$. Furthermore, it maintains these properties even when composed in parallel with another AWQSS instance using the same keys $k_D$ (but independent $\vec{x}_D$).

Sketch. First consider the case where the dealer $D$ is honest. To see Share termination, observe that every honest party eventually receives a share of $|0\rangle$ and $D$ will pass the zero purity test. It can then complete the teleportation. An honest reconstructor $R$ receives the authentication keys from the TTP, so it will only reconstruct using the original authenticated shares. Furthermore, every honest party will eventually send an authenticated share to $R$, so $R$ will receive enough authenticated shares to reconstruct. Crucially, this includes the honest parties which received their shares too late to participate in the initial zero purity test.

Now suppose that the dealer $D$ is dishonest. Using the zero purity test, we can show that the shares held by honest parties uniquely determine a value $s'$. In particular, this includes the shares held by honest parties which were unable to participate in the initial zero purity test. The late checking mechanism allows these parties to check their shares against the shares used in the initial zero purity test. The $n - t$ authenticated shares used by an honest reconstructor $R$ will contain at least $n - 2t > \lfloor \frac{n}{4} \rfloor$ honest shares. These uniquely determine the non-$\perp$ value which $R$ can reconstruct. Share termination with a dishonest dealer uses the fact that the TTP notifies the parties when the zero purity test has completed. We can show that if the TTP does this, then all parties have received a share and will eventually receive a zero purity test result from the TTP. Then they can terminate.

See the full version of the paper for more details.

5.3 Asynchronous Verifiable Quantum Secret Sharing

An asynchronous verifiable quantum secret sharing (AVQSS) scheme is described by a pair of protocols $(\text{Share}, \text{Reconstruct})$. In the first protocol, Share, a designated party called the dealer $D$ distributes a private input $s$ among the set of parties. In the second protocol, Reconstruct, the parties jointly participate and a designated receiver $R$ obtains the secret.

Definition 7 (Asynchronous verifiable quantum secret sharing). Let $(\text{Share}, \text{Reconstruct})$ be a pair of protocols for $n$ parties, where a designated party $D$, called the dealer, has a private input quantum state $s$ for Share, and each honest party that completes Share subsequently invokes Reconstruct with its local output of Share, and a designated receiver $R$ outputs a quantum state upon terminating. We say that $(\text{Share}, \text{Reconstruct})$ is a $(t, \varepsilon)$-secure asynchronous verifiable quantum secret sharing scheme if the following holds with probability at least $1 - \varepsilon$ whenever up to $t$ parties are corrupted:

- Termination:
  - If $D$ is honest, then every party eventually terminates Share.
If some honest party terminates \texttt{Share}, then all honest parties eventually terminate \texttt{Share}.

If all honest parties terminate \texttt{Share} and start \texttt{Reconstruct}, then (an honest) \texttt{R} eventually outputs a quantum state.

Privacy: If \texttt{D} is honest, the view of the adversary is independent of \texttt{s}.

Correctness: If \texttt{D} is honest, then if \texttt{R} outputs a state at the end of \texttt{Reconstruct}, the state is \texttt{s}.

Commitment: Even if \texttt{D} is corrupted, if all honest parties terminate \texttt{Share}, their joint views defines a unique state \texttt{s}', such that if \texttt{R} outputs a state after \texttt{Reconstruct}, the state has \texttt{neg}(\kappa) trace distance to \texttt{s}'.

The protocol is proceeds in two levels. In the first level, the secret is shared using AWQSS, and then in the second level each share is shared again with AWQSS. This allows the sharing scheme to be such that corrupted parties cannot arbitrarily contribute wrong shares during the reconstruction step; they can only refuse to contribute shares. In contrast to the protocol for AWQSS, we will define a fixed set of size at least \(n - t\) parties that will contribute shares during the reconstruction. Within this set there are at least \(n - 2t > n/2\) honest parties, and, because corrupted parties cannot contribute wrong shares, the reconstructor \texttt{R} can safely recover the secret.

\begin{table}[h]
\centering
\begin{tabular}{|c|}
\hline
\textbf{Functionality AVQSS Classical TTP} \\
\hline
\textbf{Initialization step.} The TTP receives keys \((k, \vec{x})\) from the dealer \texttt{D} and keys \((k_i, \vec{x}_i)\) from each party \texttt{P}_i. It sets \(S = \emptyset\), sets \(\text{counts}[i] = 0\) for \(i = 1, \ldots, n\), and sets \(S_{\text{purity}} = \emptyset\).

\textbf{WSS Level-One Execution} Upon receiving \((\text{WSS1Complete}, j)\) from a player \texttt{P}_i for the first time, the TTP does the following:

1: if \(|S| < n - t\) and \(j \notin S\) then 
2: \(\text{counts}[j] \leftarrow \text{counts}[j] + 1\)
3: \(\text{if} \ \text{counts}[j] = t + 1, \ \text{set} \ S \leftarrow S \cup \{j\}. \ \text{Then, if} \ |S| = n - t, \ \text{send} \ (\text{WSS1}, S) \ \text{to all parties.}\)
4: end if

\textbf{Zero Purity Test} Upon receiving \((\text{AQVSSShareCheck}, m)\) from \texttt{P}_i, where \(m\) is a set of measurements and \texttt{P}_i has not sent a \text{AQVSSShareCheck} message before, the TTP does the following:

1: \ Verify the authentication codes on \(m\) using the classical authentication keys. If it fails, discard this message. Otherwise, continue:
2: if \(|S_{\text{purity}}| < n - t\) then
3: \(S_{\text{purity}} \leftarrow S_{\text{purity}} \cup \{i\}\)
4: \ If \(|S_{\text{purity}}| = n - t\), perform the zero purity test, storing the result in \texttt{result}. Then send \((\text{AQVSSShareCheck}, \texttt{result})\) to each \(P_j\) for \(j \in S\).
5: else
6: \ Perform the purity test \(s\) times in the computational and Fourier bases using \(m\).
7: \ Send \((\text{AQVSSShareCheck}, \texttt{acc})\) to \(P_i\) if this test accepts and if \texttt{result} = \texttt{acc}. Otherwise send the message \((\text{AQVSSShareCheck}, \texttt{rej})\) to \(P_i\).
8: end if
\end{tabular}
\end{table}

Teleportation and Sharing Termination Upon receiving teleportation measurements from the dealer \(D\), the TTP does the following:

1. Applies the measurements to complete the teleportation by transforming the classical authentication keys.
2. Remove the dealer’s level-one AWQSS authentication keys by modifying the level-two AWQSS authentication keys.
3. Inform all parties that the sharing is over.

Reconstruction The TTP sets \(S_{\text{recon}} = \emptyset\), then participates in \(R\)’s VSS execution to share half an EPR pair.

Upon receiving \((\text{VSSReconstruct}, m_i)\) from party \(P_i\), where \(m_i\) are measurements, the TTP does the following:

1. Update the keys \(\vec{x}_i\) according to the transversally-applied teleportation circuit.
2. Attempt to reconstruct the level-one shares of the teleportation measurements using \(m_i\) (which consists of the level-two shares of the teleportation measurements).
3. for Each new successful reconstruction of \(P_j\)’s level-two sharing do
4. Set \(S_{\text{recon}} \leftarrow S_{\text{recon}} \cup \{j\}\)
5. Save the reconstructed level-one share (which \(P_j\) shared as its level-two value).
6. end for
7. if \(|S_{\text{recon}}| \geq \frac{n}{2}\) then
8. Use the reconstructed level-one shares in \(S_{\text{recon}}\) to reconstruct the measurement result of the teleportation circuit.
9. Send the reconstructed teleportation measurement result to \(R\).
10. end if

We describe the protocol from the point of view of party \(P_i\). The dealer \(D\) aims to share the contents of a register \(R_D\) with the other players.

Initialization step. \(D\) chooses a random key \(k_D\). It generates \(2s + 2 |0\rangle\) states for \(s = \text{poly}(\kappa)\) and encodes them using quantum Shamir secret sharing with threshold \(\lfloor \frac{n}{2} \rfloor + 1\) and \(n\) shares. For each encoded \(|0\rangle\) state, send the \(i\)-th share to \(P_i\) using the key \(k_D\) and freshly random \(x\) in the Verified Authentication protocol. Let \(\vec{x}\) be the vector containing each \(x\) which is used. The dealer \(D\) sends the keys \((k_D, \vec{x})\) to the TTP.

Sharing Execution.

1. Let \(R_1, \ldots, R_{2s+2}\) be the registers containing the (authenticated) shares which \(P_i\) received from \(D\). Share each state using AWQSS and key \(k_i\). Let \(R_{i,j}\) denote the register containing the \(j\)-th share.
2. For each sharing that terminated from party \(P_j\), send \((\text{WSS1Complete}, j)\) to the TTP.
3. Upon receiving \((\text{WSS1}, S)\) from the TTP, wait until all the sharings in \(S\) terminate.
4. Perform the measurements for the zero purity test on the shares from parties in \(S\).
5. Store the measurements in \(m\).
6. Send \((\text{AVQSSShareCheck}, m)\) to the TTP.
7. Wait to receive \((\text{AQVSSShareCheck}, \text{result})\) from the TTP.
8. if result is acc then
8: Generate an EPR pair share using the two remaining shares. Send the share of
the first half of the EPR pair to the dealer.
9: else
10: Abort
11: end if

Teleportation
1: $D$ waits until it receives $\left\lfloor \frac{n}{2} \right\rfloor + 1$ correctly authenticated shares from the parties, then
uses them to reconstruct the EPR pair half (this is possible since it knows the keys
and can update them according to the EPR construction, which is deterministic).
2: $D$ uses the reconstructed EPR pair half to teleport the contents of $R_D$, sending the
measurement results to the TTP.
3: The TTP informs all parties that the sharing is over.

Reconstruct
1: The reconstructor $R$ shares half of an EPR pair using AVQSS, where the other parties
use the same key as before.
2: Each party $P_i$ transversally teleports the state shared by $D$ to $R$ using $R$’s shared
EPR pair. They send the (authenticated) measurements to the TTP.
3: $R$ receives the teleportation measurements from the TTP and applies them to finish
the teleportation.

Lemma 8. Protocol AVQSS is a ($t$, neg($\kappa$))-secure asynchronous verifiable quantum secret
sharing scheme for all $t < \frac{n}{4}$.
Furthermore, it maintains these properties even when composed
in parallel with another AVQSS instance using the same keys $k_D$ and $k_i$ for each party $P_i$ (but independent $\vec{x}_D$, $\vec{x}_i$).

Proof. First, note that the AWQSS scheme satisfies termination, privacy, correctness, and
commitment with all but neg($\kappa$) probability by Lemma 6 since $t < \frac{n}{4}$. By union bound,
these properties hold for all of the AWQSS instances with all but neg($\kappa$) probability, so in
the following we condition on these properties holding for all instances.

Furthermore, these properties hold even if AWQSS is composed in parallel with another
AWQSS instance using the same dealer key $k_D$, by Lemma 6. A similar statement applies for
the zero purity test and Verified Authentication of Zeros protocol (see Lemma 4). Additionally,
the classical TTP and (ideal) authentication codes compose in parallel with themselves.

We begin by considering the case where the dealer $D$ is honest and show that all honest
parties eventually terminate Share. We first prove that the TTP will eventually send a
level-two AWQSS sharing session identifier ($WSS_1$, $S$) to all parties, then show that all honest
parties terminate every sharing session in $S$. This is sufficient to show termination since
every honest party which terminates all sharing sessions in $S$ will participate in the $|0\rangle$ purity
test. This implies that the TTP will receive at least $n - t$ measurements for this test and
will send the results to all parties. At this point, all honest parties terminate.

To see that the TTP will eventually send a level-two sharing session identifier ($WSS_1$, $S$)
to all parties, first observe that every honest party terminates the initial AWQSS sharing
phase by the properties of AWQSS since $D$ is honest. Then, in the second level of sharing
every honest party terminates the AWQSS sharing phase for each honest level-two dealer (of
which there are $n - t$, since all completed the level-one sharing). Therefore, the TTP receives
confirmation of termination of the level-two sharings from every honest party (total $n - t$)
for every honest level-two sharing (total $n - t$). This is sufficient to trigger the thresholds
for the TTP to decide and send $S$. Finally, to show honest termination of every session in
S, note that the TTP receives more confirmations of termination \((t + 1)\) for each session in \(S\) than there are corrupted parties \((t)\). Therefore, for each session in \(S\) some honest party terminated that session, and so by the properties of AWQSS all honest parties eventually terminate that session.

Privacy with an honest dealer follows naturally from the privacy of the underlying AWQSS. The honest parties all share the level-one AWQSS shares they receive from the dealer in their level-two AWQSS. Let \(P_i\) be such an honest party, who receives share \(s_{ih}\) from the original dealer in the level-one AWQSS. By the privacy guarantee of the level-two AWQSS, the adversary obtains no information about \(s_{ih}\). Hence, the privacy of the overall AVQSS scheme follows from the privacy of the level-one AWQSS.

Correctness with an honest dealer follows from commitment with a corrupt dealer, which we prove later, as well as correctness of the underlying AWQSS. Since the honest parties share the level-one shares as their level-two secrets, the joint view of the honest parties defines the unique reconstruction value guaranteed by commitment to be the original secret shared by the honest dealer.

Now consider the case of a corrupt dealer. To show sharing termination, observe that if some honest party terminates \(Share\), then it received both a zero purity test result and a set \(S\) from the TTP. Both of these are eventually received by all honest parties, since the TTP sends them to all parties. The only other place an honest party might hang is while waiting for all level-two AWQSS in \(S\) to complete. Since the TTP constructs \(S\) to contain level-two AWQSS sessions which more than \(t\) parties have reported termination on, for each session in \(S\) it holds that some honest party must have terminated that session. Therefore, all other honest parties will terminate each session in \(S\) as well.

To show reconstruction termination with a corrupt dealer, first recall that if some honest party starts \(Reconstruct\), it must have terminated \(Share\), and so all honest parties will eventually terminate \(Share\). If \(R\) is honest, then all honest parties terminate the AVQSS step in reconstruction where \(R\) shares half an EPR pair. All honest parties will perform the teleportation step, so the TTP will receive enough authenticated measurements to complete the teleportation.

Finally, we show commitment regardless of whether or not the dealer is corrupted. To do this, we show that the TTP will successfully reconstruct precisely the teleportation measurement which would have been made if the teleportation circuit was not evaluated transversally. By the commitment property of the underlying AWQSS protocol, it holds that for each level-two sharing by \(P_i\) there is a unique reconstruction state \(s_i'\) such that the TTP either reconstructs \(s_i'\) or fails to reconstruct \(P_i\’s\) level-two sharing.\(^5\) Furthermore, since the zero purity test accepted, these values \(s_i'\) must be consistent shares of the same secret teleportation measurement \(s_D'\) except with probability \(\neg\kappa(\kappa)\) (recall that the zero purity test establishes that these encode a state within small trace distance of \(|0\rangle\), which are then transformed together by transversal operations). The measurement translates the guarantee of low trace given by the zero purity test into a small failure probability. In this case, there is a unique measurement \(s_D'\) which the TTP can reconstruct. By the correctness of the level-two AWQSS sharings, the TTP will at a minimum be able to reconstruct all level-two secrets dealt by honest parties, which corresponds to the level-one AWQSS shares they hold. Since the collection of level-one AWQSS shares held by honest parties uniquely determines the dealer’s secret teleportation measurement \(s_D'\), the TTP can therefore successfully reconstruct

\(^5\) After performing the teleportation circuit, the teleportation measurement and the shared value are identical and are classical.
s_D using only the reconstructed level-two secrets dealt by honest parties, except with \( \text{neg}(\kappa) \) probability. The low trace distance guarantee again translates into a small failure probability since the shares are measured. The successful reconstruction by \( R \) then follows from the correctness of the teleportation circuit.

### 5.4 Asynchronous Toffoli Gate Computation

As discussed in the full version, the secret-sharing and authentication schemes allow for transversal evaluation of Clifford operations on the secret. In order to allow Toffoli operations, which form a universal gate set together with Clifford operations, the parties will also share a set of Toffoli states. Toffoli gates can be performed using Clifford operations and an ancillary Toffoli state [8, Appendix F]. The shared Toffoli state generation protocol constructs one or more (for sake of exposition, we describe the single version) Toffoli states which are shared amongst the parties.

At a high level, the shared Toffoli state generation protocol requires each player to send a set of Toffoli states. These sets are then tested for polynomial closeness to Toffoli states using quantum tomography and cut-and-choose techniques. This results in a set of states which are polynomially close to a set of Toffoli states with respect to trace distance. Finally, one of the sets which passes the test is further purified to achieve an exponentially good Toffoli state using techniques from fault-tolerant quantum computation [2]. Details can be found in the full version of the paper.

**Lemma 9.** For all \( t < \frac{n}{4} \), with all but negligible probability every honest party terminates the Toffoli Sharing protocol and holds a share of a state with \( \text{neg}(\kappa) \) trace distance from a Toffoli State.

### 5.5 Asynchronous Multiparty Quantum Computation

We combine the previous tools together to construct an asynchronous multiparty quantum computation (AMQPC) protocol. At a high level, the parties first construct a set of secret-shared Toffoli states which will later allow Toffoli gates to be performed on secret-shared states. They provide their inputs via AVQSS and use the classical TTP to aid in deciding a core-set of inputs. Using the shared Toffoli states, they transversally evaluate the circuit on the selected inputs. Finally, they reconstruct the states on the output wire(s) for each party.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>AMPQC Classical TTP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialization step.</strong></td>
<td>The TTP receives keys ((k_i, x_i)) from each party (P_i). It sets (S = \emptyset) and sets (\text{counts}[i] = 0) for (i = 1, \ldots, n).</td>
</tr>
<tr>
<td><strong>Core-Set Agreement</strong></td>
<td>Upon receiving a new message ((\text{Core-Set}, j)) from a player (P_i), the TTP does the following:</td>
</tr>
<tr>
<td>1: if (</td>
<td>S</td>
</tr>
<tr>
<td>2: (\text{counts}[j] \leftarrow \text{counts}[j] + 1)</td>
<td></td>
</tr>
<tr>
<td>3: if (\text{counts}[j] = t + 1) then</td>
<td></td>
</tr>
<tr>
<td>4: (S \leftarrow S \cup {j})</td>
<td></td>
</tr>
<tr>
<td>5: if (</td>
<td>S</td>
</tr>
<tr>
<td>6: Send ((\text{Core-Set}, S)) to all parties.</td>
<td></td>
</tr>
<tr>
<td>7: end if</td>
<td></td>
</tr>
</tbody>
</table>
Output For each participant $P_j$, participate in the reconstruction of $P_j$’s output wire with $P_j$ as the reconstructor.

**Protocol AMPQC**

**Initialization** Each party $P_i$ sends its authentication keys $(k_i, \vec{x}_i)$ to the TTP.

**Execution**
1. $P_i$ participates in the generation of shared Toffoli states protocol.
2. $P_i$ shares its input using VSS and simultaneously acts as a receiver in the other VSS instances. In the other VSS instances, $P_i$ uses the same authentication key $k_i$ as it sent to the TTP. The other part of the authentication key for each instance is according to $\vec{x}_i$ and is distinct for each VSS instance.
3. For each sharing that terminated from party $P_j$, send $(\text{Core-Set}, j)$ to the TTP.
4. Upon receiving $(\text{Core-Set}, S)$ from the TTP, wait until all sharings in $S$ terminate.
5. Evaluate the circuit $C$ transversally on the shares of inputs from $S$ with the help of the TTP.
6. for Each party $P_j$ do
7. Participate in the reconstruction of $P_j$’s output wire, where $P_j$ is the reconstructor.
8. end for
9. $P_i$ outputs the result of reconstructing $P_i$’s output wire.

**Theorem 10.** Protocol AMPQC $t$-securely computes any circuit $C$ for all $t < \frac{n}{4}$.

**Sketch.** At a high level, the simulator will participate honestly in most of the protocol, using dummy inputs for the honest parties. It will reconstruct the adversarial inputs using the honest parties’ shares and send these to the ideal functionality. Finally, during the reconstruction of output wires step, it will force corrupted parties to reconstruct their prescribed outputs by using the honest majority to force teleportation of the output state received from the ideal functionality (note that this state is not on the output wire). See the full version for more details.

**References**


Asynchronous Multi-Party Quantum Computation


