

I Prove, Therefore I Am: Spatiotemporal Multi-Party Computation

Ziqing Guo¹, Fuyuki Kitagawa², and Xiao Liang^{3*}

¹ The Chinese University of Hong Kong, Hong Kong
`ziqing-guo@link.cuhk.edu.hk`

² NTT Social Informatics Laboratories, Japan
`fuyuki.kitagawa@ntt.com`

³ The Chinese University of Hong Kong, Hong Kong
`xiaoliang@cuhk.edu.hk`

Abstract. Secure multiparty computation (MPC) enables mutually distrustful parties to compute on private digital inputs. We initiate the study of *spatiotemporal MPC*, extending this paradigm to functionalities whose inputs additionally depend on physical facts such as the parties’ locations, times, or trajectories. Such protocols must simultaneously hide spatiotemporal information and ensure its physical consistency: a malicious party should not be able to make the functionality operate on a spatiotemporal input inconsistent with its actual physical state.

The main conceptual challenge is to formulate extraction of spatiotemporal information within the simulation-based security framework. We introduce *arguments of spatiotemporal knowledge*, following the principle “I prove, therefore I am:” rather than defining physical presence directly through a mathematical relation, we define it operationally through the ability to complete a sound spatiotemporal verification protocol. Accordingly, an extractor recovers a spatiotemporal point from a successful prover and certifies its physical validity by using the extracted prover to succeed in an auxiliary spatiotemporal verification protocol. Building on this notion, we define universally composable (UC) security for spatiotemporal MPC, capturing privacy, physical consistency, and composability.

We provide constructions achieving this new MPC notion. We first construct UC-secure commit-and-prove protocols for spatiotemporal knowledge: in the CRS model under LWE against quantum provers without pre-shared entanglement, and in the QROM against quantum provers with unbounded pre-shared entanglement. Using these protocols, we obtain general-purpose UC-secure spatiotemporal MPC from semi-honest post-quantum MPC. We also extend our framework to UC-secure spatiotemporal multiparty quantum computation over private quantum inputs and classical spatiotemporal information.

Keywords: Secure Multi-Party Computation · Position Verification · Universal Composability · Quantum Cryptography · Proof of Knowledge

* The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No.: CUHK 24208025].

Table of Contents

Abstract	i
Table of Contents	ii
1 Introduction	1
1.1 Our Results	4
1.1.1 New Definitions: I Prove, Therefore I Am	4
1.1.2 New Constructions	5
1.2 Extensions and Future Directions	7
1.3 Related Works	9
2 Technical Overview	10
2.1 Simulation with Physically Meaningful Inputs	10
2.2 Commit-and-Prove of Spatiotemporal Knowledge	12
2.3 From Commit-and-Prove to Spatiotemporal MPC	15
2.4 Lifting the Construction to Quantum Functionalities	16
3 Preliminaries	18
3.1 Post-Quantum Extractable and Commitment scheme	18
3.2 Post-Quantum Zero-Knowledge Arguments	20
3.3 UC Framework	20
4 Spatiotemporal Verification and Commitment	21
4.1 The Model	21
4.2 Verification of Position and Spatiotemporal Trajectory	23
4.3 Spatiotemporal Trajectory Commitment	24
4.3.1 Definition	24
4.3.2 Spatial Dimension as a Barrier to Privacy	27
4.3.3 Construction	28
5 UC-secure Spatiotemporal Multi-party Computation	33
5.1 The SUC Framework	33
5.2 SUC Spatiotemporal Commit-and-Prove	37
5.2.1 Definition	37
5.2.2 Construction	38
5.3 Construction: Spatiotemporal MPC	41
6 Spatiotemporal Multiparty Quantum Computation	46
7 Acknowledgements and AI Disclosure	46
References	47

1 Introduction

A central theme in modern cryptography is how to compute while preserving privacy. This theme is epitomized by *Secure Multiparty Computation* (MPC) [Yao82, Yao86, GMW87], which allows a set of mutually distrustful parties to jointly evaluate a function on their private inputs while revealing no information beyond what is implied by the output. In this sense, MPC offers an idealized form of private computation for essentially any efficiently computable functionality. In recent decades, advances in both the theoretical and practical foundations of MPC have brought us closer to realizing this vision for a broad range of *conventional* computational tasks.

The scope of what we regard as “computation,” however, has expanded together with advances in science and technology. Computation is no longer confined to evaluating mathematical functions on explicitly represented digital inputs. For example, *position-verification* protocols [CGMO09] seek to certify that a prover is located at a specified geographic position; *physical zero-knowledge* protocols [FFN14] allow a prover to establish physical properties (such as the equality of neutron radiographs or properties of a DNA profile) without revealing the physical information underlying those properties; *proof of quantumness* [BCM⁺18] allows an efficient classical verifier to certify that a device possesses genuinely quantum computational capabilities; *verification of quantum depth* [CH22] allows a classical verifier to certify that a device performed a computation having sufficiently large quantum circuit depth. In each of these examples, the task of interest depends not only on abstract data, but also on a physical object or a physical property of a participating device. This broader perspective naturally raises the following question:

Can the notion of MPC be extended to support the secure computation of physical functionalities?

Spatiotemporal MPC. In this work, we focus on a particular class of physical functionalities, which we call *spatiotemporal functionalities*. Informally, consider parties $P_1, \dots, P_n$, where each P_i holds an ordinary private input x_i and is associated with spatiotemporal information ℓ_i , such as its physical location at a specified time or, more generally, a trajectory over an interval of time. The parties wish to jointly evaluate a functionality $F((x_1, \ell_1), \dots, (x_n, \ell_n))$ without revealing their inputs or spatiotemporal information beyond what follows from the prescribed output.

A protocol for *spatiotemporal MPC* should provide guarantees analogous to those of traditional MPC, together with an additional form of *physical consistency*. In particular, the private inputs x_i and spatiotemporal data ℓ_i should remain hidden, while a corrupted party should be unable to make the functionality operate on spatiotemporal information that is inconsistent with its physical reality. Also, we emphasize the *general-purpose* nature of this objective. Our goal is not merely to design a specialized location-verification mechanism for each individual application. Rather, as in general-purpose MPC, we seek a framework capable of securely realizing arbitrary functionalities whose inputs may include physical locations, times, or trajectories.

We next give three motivating examples:

1. **Privacy-preserving regional analytics.** Consider a service provider (e.g., an LLM provider) that offers different subscription plans across geographic regions. The provider, payment processors, and regional partners may wish to compute statistics such as regional conversion rates, demand estimates, fraud rates, or candidate price tiers. In the notation above, the ordinary inputs x_i include subscription attributes and payment records, while the spatiotemporal inputs ℓ_i describe the participating devices’ locations at the relevant transaction times. Traditional MPC can protect the privacy of the ordinary inputs. However, merely supplying a geographic

region as another ordinary input does not establish that it corresponds to the participant’s actual location. Spatiotemporal MPC adds precisely this physical-consistency guarantee while preserving the input privacy of MPC: the computation uses locations consistent with the participants’ physical behavior, yet hides both the ordinary records and the underlying spatiotemporal information beyond what follows from the aggregate output.

2. **Anonymous voting with geography-dependent representation.** Geographic information can also determine how votes are aggregated or translated into representation. In the U.S. Electoral College, each state is allocated a number of electors equal to the size of its congressional delegation, and, with the exceptions of Maine and Nebraska, states use a winner-take-all rule for appointing electors.⁴ Similarly, seats in the European Parliament are allocated among Member States according to *degressive proportionality*: more populous states elect more representatives, while less populous states receive more seats per capita.⁵ These examples motivate voting functionalities in which the rule applied to a ballot depends on the voter’s jurisdiction. Spatiotemporal MPC could allow voters to submit secret ballots while privately binding each ballot to the correct district or region. The protocol would then apply the appropriate aggregation or weighting rule without revealing voters’ identities, precise addresses, or districts beyond what is implied by the final result. Such a mechanism could also be useful in shareholder voting, union governance, digital cooperatives, or cross-regional online communities that seek geographically structured representation while retaining ballot secrecy.
3. **Collaborative augmented reality.** Co-located augmented-reality (AR) systems integrate virtual objects into a shared physical environment in real time [Azu97]. Collaborative AR additionally requires multiple users to observe and interact with a consistent collection of virtual objects [BK02]. Participating devices may wish to jointly compute object placement, scene alignment, occlusion, and interaction updates from ordinary inputs x_i , such as camera data, spatial maps, and user interactions, together with spatiotemporal inputs ℓ_i describing their physical locations and trajectories. Although traditional MPC can keep the supplied data private, it cannot by itself prevent a malicious device from manipulating the shared scene through a fictitious location or trajectory. Our spatiotemporal extension addresses this gap by binding the locations and trajectories used in these updates to the devices’ actual physical behavior. Crucially, this binding does not come at the expense of privacy: both the ordinary data and the spatiotemporal information remain hidden except for what is revealed by the prescribed scene updates.

These examples are only illustrative. Other potential applications include privacy-preserving proximity services, location-dependent access control, collaborative robotics, traffic and mobility analytics, disaster response, contact tracing, and the enforcement of geographic or temporal constraints on distributed computations. Rather than treating each such application in isolation, spatiotemporal MPC seeks a common cryptographic abstraction for computations whose correctness depends on spatiotemporal facts.

Wait! Why Not Position Verification? A vigilant reader may observe that there is already a closely related notion known as *position verification* (PV) [CGMO09, BCF⁺11, BCF⁺14]. As mentioned earlier, a PV protocol allows a prover to convince a (collection of) verifier(s) that the prover is located at a designated position. However, PV alone does not suffice for our intended notion of spatiotemporal MPC. A standard PV protocol merely certifies the prover’s asserted position; it does not enable multiple parties to evaluate an arbitrary functionality of their respective inputs and

⁴ <https://www.archives.gov/electoral-college/allocation>

⁵ <https://www.europarl.europa.eu/news/en/faq/4/how-are-meps-elected>

physical states. Also, it does not provide the input-privacy guarantee required by general-purpose MPC: the asserted position is ordinarily part of the public statement, and the prover and verifiers do not have the additional private inputs that appear in an MPC execution.

Recent work of Girish et al. [GGG⁺26] introduces a zero-knowledge variant of position verification, allowing a prover to establish its location while hiding all information not implied by this fact. This represents an important step toward privacy-preserving proofs of spatiotemporal facts. Nevertheless, zero-knowledge PV remains a proof functionality rather than a general-purpose computation functionality. Moreover, the soundness guarantee in [GGG⁺26] is seemingly relaxed: a prover can convince the verifier that it is located within a specified (large) region without revealing its exact position within that region. While this suffices for certain applications, it does not support finer-grained spatiotemporal functionalities in which the prover must prove that it is at a specific point in space-time.

In fact, this should not be treated as a drawback of [GGG⁺26]; it is due to an inherent distinction between hiding a position and computing on it. If the public statement identifies an exact point in space-time, then that point is necessarily revealed by the statement itself and cannot simultaneously be hidden by zero knowledge. Conversely, a statement asserting membership in a larger region may conceal the exact point, but does not *by itself* enable a functionality whose output depends on that exact point. Spatiotemporal MPC must reconcile these requirements by allowing the functionality to use fine-grained spatiotemporal information internally while hiding it from the other parties, except insofar as it is revealed by the output. Thus, a substantive gap remains between existing notions of position verification and our desired notion of spatiotemporal MPC.

Spatiotemporal Knowledge Extraction. The aforementioned gap directs attention to a critical difference between proof/argument systems and general-purpose MPC: the role of *input extraction* (and *arguments of knowledge*). The security of traditional MPC is typically formulated through the real/ideal simulation paradigm. In the ideal world, each party submits an input to a trusted functionality, which evaluates the desired function and returns the prescribed outputs. Security requires that every real-world adversary can be simulated by an ideal-world adversary, so that the real execution reveals no more and permits no greater influence than the ideal execution. For a corrupted party, the input supplied to the ideal functionality need not coincide with the input that the party was nominally instructed to use. A malicious party may deviate from the protocol and thereby use some other *effective input*. The simulator must therefore determine (or *extract*) the effective input on which the corrupted party’s behavior is based, and submit that input to the ideal functionality. Without such an extracted input, there is no well-defined ideal-world computation against which the real execution can be compared.

For ordinary MPC, the extracted object is mathematical: a bit string, a field element, an **NP** witness, or some other explicitly represented value. Spatiotemporal MPC requires something conceptually different. A corrupted party’s relevant input may include its spatiotemporal property. To formulate simulation security, we must explain what it means for a simulator to *extract such a property* and how the extracted object is related to the party’s physical reality.

To the best of our knowledge, no existing work provides the corresponding notion of *spatiotemporal knowledge extraction*.⁶ This issue is not merely definitional. A party may move during the execution of a protocol, relay messages through agents at different locations, or behave in a manner that is not naturally associated with a single point in space-time. A satisfactory solution must determine when such behavior corresponds to a well-defined spatiotemporal input, what an extractor

⁶ A concurrent work by Bartusek et al. [BHOY26] introduces a related definition. We provide more information in Sec. 1.3.

should output when it does not, and which physical resources or observations the extractor is permitted to use. These questions have no direct analogue when the target of extraction is an ordinary mathematical witness.

In summary, spatiotemporal MPC presents new conceptual and technical challenges, and calls for a formal treatment of conventional cryptographic guarantees in a physically-consistent manner.

1.1 Our Results

We initiate a systematic study of spatiotemporal MPC and develop a collection of definitions and constructions that lay the foundation for this new notion. We summarize our main results below.

1.1.1 New Definitions: I Prove, Therefore I Am

Arguments of Spatiotemporal Knowledge. As discussed above, the main conceptual gap between position verification and spatiotemporal MPC is the absence of an appropriate notion of *extraction* for spatiotemporal information. As our first conceptual contribution, we introduce a new paradigm for overcoming this definitional obstacle.

Our approach is guided by the following philosophy: it is challenging to define a physical property using mathematical logic. We first examine successful examples of defining physical properties, such as proofs of quantumness and position verification, and observe that they all rely on the notion of a “proof.” This observation suggests a simple reversal of perspective: why should we undertake the challenging task of capturing a physical property using mathematical logic, *when the interactive protocol used to test that property can itself serve as its definition?* In other words, we can directly use the prover’s **ability** to pass position verification as the **definition** of its physical presence at the asserted position.⁷ Inspired by this perspective, we propose the following notion of an *argument of spatiotemporal knowledge*:

Definition 1 (Informal). *A position-verification protocol $\Pi = \langle P, V \rangle$ is an argument of spatiotemporal knowledge if there exist an efficient two-stage extractor $\text{Ext} = (\text{Ext}_0, \text{Ext}_1)$ and a sound position-verification protocol $\Pi' = \langle P', V' \rangle$ with the following property:*

- *For every potentially malicious efficient prover P^* , if P^* convinces V with probability δ , then $\text{Ext}_0^{P^*}$ extracts a spatiotemporal point st^* together with auxiliary information aux such that $\text{Ext}_1^{P^*}(\text{st}^*, \text{aux})$ can act as the prover in Π' and convince V' that it is located at st^* with probability at least $\text{poly}(\delta) - \text{negl}(\lambda)$.*

A few aspects of this definition merit further clarification. First, observe that the second position-verification protocol Π' is existentially quantified. Thus, satisfying Def. 1 does not require $\text{Ext}_1^{P^*}(\text{st}^*, \text{aux})$ to succeed in every possible position-verification protocol. It suffices for it to succeed in a single protocol Π' , provided that Π' satisfies the standard soundness requirement for position verification. This choice reflects the role of Π' in the definition: it is used only to certify the physical validity of the extracted spatiotemporal information st^* . If st^* were not P^* ’s real location, then $\text{Ext}_1^{P^*}(\text{st}^*, \text{aux})$ could not successfully complete *any* sound position-verification protocol. Taking the contrapositive, successful participation in *some* sound protocol Π' suffices to certify the validity of st^* . This existential formulation also provides greater flexibility in designing Π and Ext , as their construction and security analysis may be tailored to a particular sound protocol Π' .

⁷ This is reminiscent of René Descartes’ famous dictum “*cogito, ergo sum*” (“I think, therefore I am”), which arose from his reflections on how to establish the existence of one’s own mind. We therefore call our framework “*I prove, therefore I am.*”

Second, [Def. 1](#) subsumes standard position-verification soundness in a straightforward manner. In a standard position-verification protocol, the claimed position is public. The extractor can therefore output this claimed position as st^* and take Π' to be the original protocol Π . Indeed, this serves as a useful sanity check on the definition. The main strength of [Def. 1](#), however, is that it captures spatiotemporal soundness in settings where the relevant position is hidden, such as zero-knowledge position verification [[GGG⁺26](#)] and the MPC setting developed in this work. In these settings, there is no publicly specified position against which the prover’s behavior can be checked. Instead, the position is implicitly determined by the behavior of the potentially malicious prover. Our definition addresses this difficulty by requiring the extractor to recover spatiotemporal information st^* from the prover’s behavior and by certifying the physical validity of st^* through the auxiliary position-verification protocol Π' .

Finally, we emphasize that [Def. 1](#) is deliberately simplified and intended only to convey the central idea behind “I prove, therefore I am.” In particular, no formal definition in the main body corresponds directly to [Def. 1](#). Rather, its operational interpretation of spatiotemporal extraction underlies the formal definitions introduced in this work (e.g., [Def. 14](#) and [15](#)). These formal definitions must address several additional subtleties, including the need to strengthen the extraction guarantees to support simulation against quantum provers. We defer discussion of these subtleties to [Sec. 2](#).

Spatiotemporal MPC. The notion of extracting spatiotemporal knowledge provides the foundation needed to define spatiotemporal MPC. We formulate our definition in the universally composable (UC) framework [[Can01](#)], extending it with our notion of arguments of spatiotemporal knowledge. This yields a definition of *UC-secure spatiotemporal MPC*. Informally, our definition captures what one would naturally expect: it allows parties to jointly evaluate a spatiotemporal functionality over their private inputs while hiding both their ordinary inputs and their spatiotemporal information. At the same time, it prevents a malicious party from supplying spatiotemporal information that is inconsistent with its actual physical state. Finally, our framework inherits UC’s general composition guarantees: UC-secure spatiotemporal MPC protocols can be used as modular components of larger UC (spatiotemporal) protocols while preserving their security guarantees.

1.1.2 New Constructions

We obtain a collection of constructions realizing our notion of UC-secure spatiotemporal MPC under standard assumptions.

Commit-and-Prove of Spatiotemporal Knowledge. Our first step is to construct a special two-party functionality with a commit-and-prove flavor. Roughly speaking, this functionality allows a party to commit to both an ordinary private input and private *spatiotemporal* information, and subsequently prove statements about the committed values without revealing them. In addition to the standard binding and zero-knowledge guarantees of commit-and-prove, the functionality guarantees that the committed spatiotemporal information is consistent with the committing party’s actual physical state. In our UC spatiotemporal framework, this consistency is captured by requiring the simulator to extract the committed spatiotemporal information and certify its physical validity through a sound spatiotemporal verification protocol. A protocol realizing this functionality plays a central role in all of our subsequent constructions. We realize it under standard assumptions.

Theorem 1 (Informal). *UC-secure commit-and-prove of spatiotemporal knowledge exists in each of the following settings:*

1. In the common reference string (CRS) model, assuming the quantum hardness of Learning with Errors (QLWE), knowledge soundness holds against quantum-polynomial-time (QPT) provers without pre-shared entanglement.
2. In the quantum random oracle model (QROM), knowledge soundness holds against QPT provers with unbounded pre-shared entanglement.

Here, pre-shared entanglement refers to entanglement distributed among the spatially separated components of a malicious prover before the protocol begins. Such entanglement constitutes a particularly powerful resource in position-verification protocols: it allows a malicious prover's components to coordinate their attacks across different locations and poses a significant obstacle to standard extraction techniques.

Spatiotemporal MPC. We next turn to general-purpose functionalities. At a high level, we use [Thm. 1](#) in a similar way that traditional commit-and-prove protocols are used in the GMW compiler [[GMW87](#), [CLOS02](#)], upgrading a semi-honest MPC protocol for traditional functionalities into a maliciously secure MPC protocol for *spatiotemporal functionalities*. The spatiotemporal nature of our setting, however, introduces several unique challenges. We discuss these challenges and our solutions in [Sec. 2](#). For now, we state our result as follows.

Theorem 2 (Informal). *Assuming the existence of semi-honest post-quantum MPC and UC-secure commit-and-prove of spatiotemporal knowledge (as provided by [Thm. 1](#)), UC-secure spatiotemporal MPC exists.*

Spatiotemporal MPQC. A promising approach to constructing position-verification protocols (and consequently, spatiotemporal MPC protocols) is to exploit the no-cloning property of quantum information (e.g., [[BCF⁺11](#), [BK11](#), [KMS11](#), [TFKW13](#), [Unr14](#), [DS21](#), [LLQ22](#), [GGG⁺26](#), [BHOY26](#)]). Since the parties in this setting are already granted quantum capabilities, it is natural to ask whether our notion of spatiotemporal MPC can be extended to support *quantum* functionalities. More specifically, we consider a setting in which each party holds a quantum input, and the parties wish to securely and jointly evaluate a quantum circuit over their respective quantum inputs and their respective classical⁸ spatiotemporal information. We call this notion *spatiotemporal multi-party quantum computation (spatiotemporal MPQC)*. We show that this is indeed possible assuming UC-secure MPQC (for traditional quantum functionalities without spatiotemporal aspects).

Theorem 3 (Informal). *Assuming the existence of post-quantum one-way functions (PQ-OWFs), UC-secure MPQC, and UC-secure zero-knowledge commit-and-prove of spatiotemporal knowledge (as provided by [Thm. 1](#)), UC-secure spatiotemporal MPQC exists.*

It follows from [[BCKM21b](#)] that UC-secure MPQC can, in turn, be constructed from UC-secure post-quantum MPC.⁹ We therefore obtain the following corollary.

⁸ Although the parties possess quantum capabilities and may hold quantum inputs, we model their spatiotemporal information classically. Quantum-mechanical notions of position, relational time, quantum clocks, and quantum reference frames have been studied e.g., [[PW83](#), [EMS⁺17](#), [GCRB19](#), [CRGBB20](#)]. However, to the best of our knowledge, these formalisms have not yet yielded a canonical operational model of quantum spatiotemporal information suitable for our cryptographic setting. We leave the formulation of a cryptographically meaningful quantum analogue to future work.

⁹ Technically, the construction also requires quantum garbled circuits. However, [[BY22](#)] shows how to construct quantum garbled circuits assuming only PQ-OWFs, which are themselves implied by the existence of post-quantum MPC.

Corollary 1. *Assuming the existence of UC-secure post-quantum MPC and UC-secure commit-and-prove of spatiotemporal knowledge, UC-secure spatiotemporal MPQC exists.*

Stand-Alone Security without CRS. Note that a CRS can be generated via a quantum-secure coin-flipping protocol. Several such protocols are known: constant-round protocols from QLWE plus QFHE [BS20, ABG⁺21, LPY23]; polynomial-round protocols from semi-honest OT [CLPY25] or PQ-OWFs plus quantum communication [BCKM21b, GLSV21]; and constant-round protocols from semi-honest OT with relaxed simulation quality¹⁰ [CCLY22a, LMS22, CLPY25]. We can instantiate the CRS in the first construction in Thm. 1 using any of these protocols to obtain a “plain-model” (i.e., without any setup such as a CRS) ZK commit-and-prove protocol for spatiotemporal knowledge, at the cost of downgrading UC security to stand-alone security.

Similarly, we can obtain plain-model versions (with stand-alone security) of Thm. 2, Thm. 3, and Corollary 1 by first implementing the UC-secure protocols in the CRS model and then replacing the CRS with one of these coin-flipping protocols. For the first step, we recall that UC-secure PQ-MPC follows from QLWE in constant rounds [PVW08, GS18]; the same holds for PQ-MPQC [BCKM21b].

1.2 Extensions and Future Directions

Our work opens several avenues for future research, some of which we highlight below.

Optimized Constructions. The constructions presented herein should be viewed as initial feasibility results, leaving substantial room for optimization, as was also the case in the development of traditional MPC. One natural goal is to obtain *black-box* constructions—that is, constructions that access their underlying building blocks only through their input/output interfaces—thereby offering greater modularity and more transparent efficiency guarantees. We believe that such constructions may be obtained by combining our ideas with the *MPC-in-the-head* techniques [IKOS07, CCLY22a].

Another important direction concerns round complexity. Our current constructions are not optimized in this respect, partly because they invoke multiple position-verification instances spanning the MPC execution to ensure that each party remains at its authenticated (hidden) position throughout the computation. For two- or three-round protocols, however, the relevant spatiotemporal interval could potentially be narrowed to the moment at which a party sends its response in the 2PC/MPC protocol. This direction is closely related to the traditional model of *non-interactive secure computation* (NISC) [IKO⁺11]. A formal treatment would require a careful analysis of the concurrent/parallel security of our constructions when composed with quantum NISC protocols, such as [Bar21, ABG26].

Deployment Potential on NISQ Devices. Recent theoretical and experimental progress suggests that quantum position verification (QPV) is approaching implementation on near-term quantum hardware [BCS22, KPB⁺25, KZG⁺26]. Also, MPC has undergone extensive optimization, resulting in increasingly practical implementations [MNPS04, BLW08, PSSW09, DPSZ12, Kel20, PKY⁺21, DEK21, WWP22, CZO⁺23]. Our constructions require little quantum capability beyond that needed to execute the underlying QPV protocol. Consequently, after suitable cryptographic and systems-level optimization, they may become candidates for implementation using NISQ-era components. Such an implementation would initiate a promising line of work at the intersection of

¹⁰ There is strong evidence that, without such relaxation, constant-round constructions are unlikely to exist [CCLY22b, CCLL26], unless one resorts to non-black-box simulation techniques, which, in the current landscape, require stronger assumptions than minimal ones such as semi-honest OT or PQ-OWF.

QPV and pragmatic MPC. In particular, it could lead to the first experimental demonstration of spatiotemporal MPC and illustrate a concrete application of near-term quantum technology.

Integration with Different PV Protocols. The constructions underlying [Thm. 1](#) use a PV protocol based on the no-cloning property of quantum information. However, our framework is more general and can be instantiated with any PV protocol satisfying the following “nice” property, adapted from [\[GGG⁺26, Definition 2.8\]](#). Roughly, we call a PV protocol *nice* if the verifiers’ joint verdict can be computed by a classical predicate applied to the classical portion of the interaction transcript, together with the relevant timing information and verifier randomness. In particular, evaluating this predicate does not require access to any residual quantum state held by the verifiers.

This observation may allow our framework to be instantiated with PV protocols based on other assumptions, including hardware models [\[GW98\]](#), bounded-retrieval models [\[CGMO09\]](#), and bounded-quantum-storage models [\[BCS22\]](#). This would yield analogues of [Thm. 1](#) and [2](#) in a variety of settings, with security inherited from the adversarial model of the underlying PV protocol. For example, instantiating our framework with a PV protocol secure against adversaries with bounded quantum memory would yield a spatiotemporal MPC protocol secure against the corresponding class of adversaries. We leave a formal treatment of these instantiations to future work.

Worth particular mention is a new notion of PV proposed by Bartusek et al. [\[BHOY26\]](#), referred to therein as *quantum localization*. Their protocols belong to the class of quantum-based PV protocols, but achieve a stronger notion of soundness. Whereas conventional soundness guarantees only that at least one member of any successful adversarial coalition is present at the claimed location, quantum localization certifies that a designated unclonable quantum state exists at the claimed spacetime point and cannot be found elsewhere. This rules out distributed strategies in which different adversaries collectively emulate a *single* prover across multiple locations or times.

It would be appealing to integrate quantum localization into our framework. Unfortunately, the verification procedure of [\[BHOY26\]](#) relies on access to a structured classical oracle in the ideal-obfuscation model. Such oracle access does not directly provide an explicit polynomial-size verification circuit: the resulting oracle-dependent acceptance predicate is not, as given, a standard **NP** relation. This presents an obstacle because our construction uses zero-knowledge proofs for **NP** relations to establish that the underlying verification procedure accepts, without revealing the private information used in that verification. These proofs therefore cannot be applied directly to the oracle-dependent acceptance predicate. Integrating the construction of [\[BHOY26\]](#) requires addressing this oracle dependence while preserving its stronger localization soundness and the privacy guarantees required by our framework. This is closely related to the open direction in [\[BHOY26, Section 1.6\]](#) of constructing trajectory commitments and zero-knowledge localization. If the oracle-dependent verification can be made compatible with our zero-knowledge machinery, while retaining the remaining properties required by our construction, this would provide a route to spatiotemporal MPC with the [\[BHOY26\]](#)-flavor localization guarantee.

Extension to General Physical Functionalities. Although this work focuses on spatiotemporal functionalities, it is natural to ask whether our framework can accommodate more general physical properties. To some extent, the framework extends to properties admitting suitable verification protocols. Consider, for example, proof of quantumness. Following the approach of [Def. 1](#), one could formulate an “argument of quantumness knowledge” and, subsequently, MPC functionalities whose inputs include a verified claim of quantumness.^{[11](#)}

¹¹ This extension may be less straightforward than it first appears. In particular, one must carefully restrict the extractor’s computational power and its access interface to P^* . We believe that potential issues can be addressed

Nevertheless, some physical properties appear to lie beyond the reach of our current framework. The difficulty stems from a subtle aspect of the extractor in [Def. 1](#): intuitively, the extractor must obtain some form of access to the adversarial prover P^* ; however, combining the extractor with P^* , may sometimes blur the boundary of the physical resource being certified and thereby yield vacuous attacks on the underlying verification protocol.

To illustrate, suppose that a proof-of-quantum-depth protocol guarantees that any successful prover must possess a quantum circuit of depth at least d . A naive definition of an “argument of quantum-depth knowledge” might give the extractor Ext_1 black-box access to P^* . This immediately creates a vacuous strategy: if P^* implements a depth- d quantum computation, then Ext_1 may invoke P^* repeatedly and sequentially, effectively assembling a quantum computation of arbitrarily large polynomial depth. The resulting extracted resource would therefore reflect the combined power of the extractor and the prover, rather than the physical capability possessed by the prover itself. This problem becomes even more pronounced for MPC functionalities involving several physical properties simultaneously, because the mechanisms used to extract one property may inadvertently supply resources relevant to another.

It remains unclear whether this difficulty is inherent in any attempt to define MPC for general physical functionalities or is merely a limitation of the framework developed in this work. Resolving this question is a very appealing, albeit ambitious, direction for future research.

1.3 Related Works

Chandran et al. [[CGMO09](#)] introduced the notion of *position-based* MPC, which supports functionalities that depend on the parties’ *public* locations. Their notion differs fundamentally from our spatiotemporal MPC: it does not hide the parties’ locations. In contrast, our notion guarantees the privacy of the parties’ spatiotemporal information while simultaneously ensuring its physical authenticity. Achieving authenticity without revealing this spatiotemporal information is, in fact, the central challenge of our work.

The concurrent and independent work of Bartusek et al. [[BHOY26](#)] also employs extraction in defining their new notion of *quantum localization*. This similarity, however, is only superficial: the two works pursue fundamentally different goals and operate in distinct settings. In their framework, extraction is used to *localize a quantum object*. Specifically, the verifier retains one share of an entangled state, while the prover holds the other; possession of the prover’s share serves as an unclonable certificate of identity. Security requires the existence of an extractor that acts only on the prover’s state at the claimed spacetime point and recovers a register entangled with the verifier’s share. Extraction thereby establishes that a particular quantum object—and, consequently, the entity identified by it—was present at the claimed spacetime point. Repeatedly applying this guarantee makes it possible to track the object’s movement and thereby recover its trajectory.

In contrast, our extractor does not recover or track an identity-bearing object located at a designated spacetime point. Rather, it plays the standard simulation-based role of identifying the effective inputs with which a malicious party participates in a computation, including its claimed spatiotemporal information. Our security definition then requires this extracted spatiotemporal information to be consistent with the party’s actual physical state. Thus, extraction in their work is itself the mechanism by which the location of a persistent quantum object is defined, whereas extraction in our work connects an adversarial protocol execution to a well-defined ideal computation whose spatiotemporal inputs must be physically authentic. Moreover, their focus is on verifying

by starting from a classical-verifier proof-of-quantumness protocol, but a detailed treatment would take us beyond the scope of the present work.

the location or trajectory of an unclonable quantum object, while ours is on securely computing general functionalities over private ordinary and spatiotemporal inputs.

2 Technical Overview

Our construction separates three requirements that are intertwined in spatiotemporal computation: extracting a party’s effective input, certifying the physical meaning of that input, and using it consistently throughout a private computation. Ordinary MPC addresses the first and third requirements, but treats every input as an abstract string. Position verification addresses the second, but ordinarily makes the position public and does not provide the extraction interface needed for simulation. Our approach connects these two forms of security through a simulation-compatible notion of “argument of spatiotemporal knowledge.”

We first explain this notion and its integration into the UC framework (Sec. 2.1). We then construct a commit-and-prove of spatiotemporal knowledge protocol (Sec. 2.2), use it to authenticate private inputs during classical computation (Sec. 2.3), and finally give a generic extension to quantum functionalities (Sec. 2.4). For exposition, we primarily consider two parties; the same ideas apply to the multiparty setting.

2.1 Simulation with Physically Meaningful Inputs

Suppose that party P_b has private input (x_b, st_b) , where initially $\text{st}_b = (L_b, T_b)$ denotes a single space-time point. Once positions and times are represented by finite strings, evaluating $F((x_0, \text{st}_0), (x_1, \text{st}_1))$ is an ordinary computational task. The difficulty is that an ordinary ideal functionality cannot distinguish an authentic position from a fabricated one. Consequently, securely computing this function on self-reported strings does not realize spatiotemporal computation.

The same distinction appears in simulation. An ordinary simulator may extract a corrupted party’s effective input (x_b^*, st_b^*) , but recovering the string st_b^* does not establish that the adversary occupied the corresponding point. We need extraction to certify not only *which value the adversary used*, but also *why that value has a valid physical interpretation*.

Operational Certification of Extracted Information. Our starting point is the “I prove, therefore I am” principle from Def. 1. Rather than require the extractor to produce a mathematical witness of physical presence, we require it to produce a strategy that succeeds in a sound position-verification protocol for the extracted claim.

At a schematic level, an extractor Ext_0 produces $(\text{st}^*, \rho) \leftarrow \text{Ext}_0^{P^*}(1^\lambda)$, and a second algorithm Ext_1 uses ρ and the prescribed access to P^* to participate in a sound position verification protocol Π' on public input st^* . Successful verification supplies the operational meaning of the extracted claim. The auxiliary protocol Π' is existentially quantified: one sound test suffices to certify the claim; success in every conceivable position-verification protocol is unnecessary.

An essential restriction concerns the extractor’s physical resources. The proving strategy must respect the causal and spatial restrictions of the adversarial model. In particular, extraction must not introduce a new agent at the claimed position or provide communication or entanglement unavailable to the adversary. Otherwise, successful verification would certify the extractor’s resources rather than the adversary’s physical presence.

Simulatable Extraction. The operational certification in Def. 1 explains why an extracted claim has physical meaning, but does not specify how the adversary can continue participating in the surrounding computation after extraction. This is an additional requirement: extracting information

may disturb the adversary’s quantum state and thereby change its subsequent behavior [GLSV21, BCKM21b, CCLY22a]. A strategy that succeeds in a separate physical verification experiment does not, by itself, resolve this problem.

We therefore strengthen operational certification with *simulation while extracting*. The first stage produces an extracted claim and a residual state, and the second stage has two alternative modes:

$$\begin{aligned} \text{Ext}_0(1^\lambda) &\longrightarrow (\text{st}^*, \rho), \\ \text{Ext}_1(\text{sim}, \text{st}^*, \rho) &\longrightarrow \text{information for a simulated continuation}, \\ \text{Ext}_1(\text{prv}, \text{st}^*, \rho) &\longrightarrow \text{a prover for } \Pi'(\text{st}^*). \end{aligned}$$

The **sim**-mode output of Ext_1 enables a continuation that is computationally indistinguishable from the real execution, while maintaining the appropriate correlations between the extracted information, the adversary’s residual state, and the surrounding environment; the **prv** mode supplies the physical certification. These roles are captured by two distinct requirements.

More concretely, consider a malicious party P^* that first commits to spatiotemporal information and then jointly evaluates a predicate on it with an honest party V , who ultimately outputs a decision bit. Let v_{real} denote the committed value in the real commit-stage execution, or $\perp$ if there is no unique valid committed value, and let ST_{P^*} denote the adversary’s residual state at the end of that stage. Our definition requires the following two properties hold simultaneously:

1. **Simulation.** Running Ext_0 followed by Ext_1 in **sim** mode, which outputs a simulated “post-extraction” state $\widetilde{\text{ST}}_{P^*}$ for the malicious P^* , as well as a simulated commit-stage acceptance bit $\widetilde{b}$ of the receiver. These outputs must satisfy:

$$(\Gamma_{\widetilde{b}}(\text{st}^*), \widetilde{\text{ST}}_{P^*}) \stackrel{c}{\approx} (v_{\text{real}}, \text{ST}_{P^*}), \quad \text{where } \Gamma_{\widetilde{b}}(\text{st}^*) := \begin{cases} \text{st}^* & \text{if } \widetilde{b} = 1, \\ \perp & \text{if } \widetilde{b} = 0. \end{cases}$$

Crucially, the comparison concerns the *joint distribution of the value and residual state*, not merely their separate distributions. Thus, the extractor must both recover the committed information and produce a simulated state with the appropriate correlations with that information. This is what allows extraction to be used within a larger computation without detectably changing the adversary’s continuation.

2. **Physical certification.** If P^* convinces V to accept with probability $\delta > \kappa(\lambda)$, where κ is the knowledge error, then the same first-stage extractor must also support a successful physical proving strategy:

$$\Pr \left[\langle \text{Ext}_1(\text{prv}, \text{st}^*, \rho), V' \rangle(1^\lambda, \text{st}^*) = \text{accept} : (\text{st}^*, \rho) \leftarrow \text{Ext}_0(1^\lambda) \right] \geq \text{poly}(\delta). \quad (2.1)$$

The proving strategy must obey the physical resource restrictions described above.

The two requirements are evaluated in separate experiments, each starting with a fresh execution of Ext_0 . They do not require copying ρ or using it for both purposes in one execution. Only **sim** mode is used to continue the simulated computation; **prv** mode is an alternative experiment certifying the physical meaning of the extracted claim. Thus, simulation mode supplies the additional guarantee needed beyond the operational intuition of Def. 1, rather than another physical test.

Integration into UC. The UC framework compares a real execution with an ideal execution in the presence of an interactive environment $\mathcal{Z}$ [Can01, CLOS02]. Crucially, $\mathcal{Z}$ can exchange information with the adversary during the execution, rather than only inspect a final transcript. The simulator must therefore support an ongoing interaction with an external machine whose state it cannot

reset. This essentially rules out rewinding-based simulation and motivates an online, straight-line interface to the environment.

Our spatiotemporal extension retains this interactive simulation requirement, with quantum machines and quantum auxiliary states. The simulator is organized as $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$. During the execution, $\mathcal{S}_0$ extracts the corrupted parties' effective ordinary and spatiotemporal inputs (encoded as a classical string) and submits them through the ideal functionality's prescribed interface. As with Ext_1 above, $\mathcal{S}_1$ has two alternative modes. Its **sim** mode continues the ideal execution so that the ordinary UC simulation requirement is satisfied: $\text{REAL}_{\Pi, \mathcal{A}, \mathcal{Z}} \approx_c \text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}$; its **prv** mode certifies the extracted spatiotemporal claims in the spirit of Eq. (2.1).

Unlike the simple predicate-evaluation setting above, a general-purpose secure computation need not have a designated verifier or an acceptance event defining δ . We therefore use the simulator's submission of an extracted input as the relevant event: for each spatiotemporal string st^* , let δ_{st^*} denote the probability that $\mathcal{S}_0$ extracts st^* and the **sim**-mode $\mathcal{S}_1$ submits it to the ideal functionality. We will also utilize a knowledge error $\kappa(\lambda)$; whenever $\delta_{\text{st}^*} > \kappa(\lambda)$, we require $\mathcal{S}_1$ in proving mode to convince the verifier of $\Pi'(\text{st}^*)$ with probability at least $\text{poly}(\delta_{\text{st}^*})$ in the corresponding certification experiment. As above, the two modes are evaluated in separate experiments, each starting with a fresh execution of $\mathcal{S}_0$. Only the simulation mode participates in the real/ideal comparison; the proving mode supplies the additional physical guarantee. Thus, physical certification supplements ordinary simulation security rather than replacing it.

On Composition. In the UC framework, the environment's ability to interact with the adversary is what makes modular composition possible: a subprotocol's simulator must remain valid when the surrounding protocol acts as its environment. Our extension preserves this interface. Consequently, the indistinguishability part of a composition proof follows the usual UC replacement argument.

The physical part requires an additional step. The composed simulator must retain the extracted claims and their associated proving strategies, and the resulting strategies must remain admissible in the physical adversarial model. In particular, replacing a cryptographic subroutine must not introduce a forbidden communication path or additional pre-shared entanglement into the position-verification reduction. Under these compatibility conditions, the same simulator composition carries the physical certificate through the replacement. This is why merely observing that our definition is stronger than UC would not, by itself, prove composition. For functionalities with no spatiotemporal inputs, the additional requirement is vacuous, recovering ordinary quantum-UC security.

From points to trajectories. The ideal functionality may equally receive a finite ordered sequence

$$\text{st}_b = ((L_{b,1}, T_{b,1}), \dots, (L_{b,m}, T_{b,m})).$$

The extraction stage now recovers this sequence, while the proving mode participates in a *spatiotemporal trajectory verification* protocol, as formalized in Def. 11. Using a trajectory-level experiment avoids silently identifying separate point-wise success probabilities with successful verification of the entire trajectory. Our trajectories specify authenticated observations, not continuous monitoring between observations. Moreover, for a distributed adversary, authenticity means that the claimed points were occupied by components of the coalition. In contrast to [BHOY26], it does not assert that a single persistent device traversed them all (see Rmk. 2).

2.2 Commit-and-Prove of Spatiotemporal Knowledge

We next show how to construct a commit-and-prove protocol for spatiotemporal predicates. Unlike a commitment to a single spacetime point, our primitive records a private *trajectory* and supports

subsequent proofs about its properties. The central challenge is to make this physical record compatible with ideal/real simulation: the simulator must extract a corrupted prover’s trajectory, certify its physical meaning, and preserve the prover’s subsequent interaction with the environment.

On the Definition. This primitive is a special case of the UC spatiotemporal 2PC definition introduced in [Sec. 2.1](#). Its ideal functionality $\mathcal{F}_{\text{CnP}}^{\text{ST}}$ has two phases. In the *Commit Phase*, it records a private trajectory st under a commitment identifier and notifies the verifier that commitment is complete. In the *Prove Phase*, it receives a public predicate f and informs the verifier whether the recorded trajectory satisfies $f(\text{st}) = 1$, without revealing the trajectory itself. Multiple commitments may accumulate over time, and a later predicate may refer to several previously recorded trajectories.

We require security of the commit phase itself, not merely of the combined commit-and-prove execution. Essentially, this phase is an *extractable-and-equivocal commitment*. The prove phase is a zero-knowledge argument of knowledge, linking the requested predicate to the same committed trajectory. Additionally, the extractability provided by the commit phase ensures that the extracted trajectory must satisfy our physical-certification requirement: the extractor’s proving mode must yield a successful strategy for a sound trajectory-verification protocol.

A basic causality constraint determines the organization of the construction. One may commit in advance to a *description* of an intended trajectory, but this does not certify that the trajectory will actually be traversed. Physical authentication must concern observations that have already occurred by the time the protocol completes.

To see the distinction, consider a purported position-verification protocol that terminates at time t_{end} and certifies that the prover will occupy position Y at a later time $t_{\text{future}} > t_{\text{end}}$. A malicious prover can always follow the honest strategy until t_{end} and subsequently arrange to occupy $Y' \neq Y$ at t_{future} . The verifier’s completed view is unchanged, contradicting soundness. Thus, a protocol cannot authenticate a freely chosen future position merely by authenticating a present intention. Consequently, committing to an authenticated trajectory requires a physical interaction spanning its observation window.

Meshing the Observation Window. Let T be a publicly specified family of admissible trajectories, represented over a polynomial-size spacetime mesh. Partition the observation window into small intervals with designated observation times

$$T_1 < \dots < T_m,$$

and let L_i be the finite set of candidate locations at time T_i . A trajectory has the form

$$\text{st} = ((L_i, T_i))_{i=1}^m, \quad L_i \in \mathsf{L}_i.$$

We schedule a position-verification instance for each candidate point in

$$S = \bigcup_{i=1}^m (\mathsf{L}_i \times \{T_i\}).$$

The commitment interval includes the transmission and processing margins needed for these instances. The mesh specifies the observations being authenticated.

Position Verification under Commitments. For simplicity, consider a two-message position-verification protocol Π_{PV} with a classical challenge q and a classical response a (e.g., [\[LLQ22\]](#)). Let Com be a post-quantum commitment that is both extractable and equivocal (such a commitment

is known, e.g., in the CRS model assuming QLWE [DFL⁺09]). At each mesh point $\alpha \in S$, the verifier sends the prescribed challenge q_α . Instead of returning a_α in the clear, the prover returns a commitment $c_\alpha = \text{Com}(a_\alpha; r_\alpha)$. Notice that hiding the response alone is not enough: the pattern of responses must not disclose the trajectory. Similar to [GGG⁺26], we use a public traffic schedule covering every candidate point. At points on its trajectory, the prover commits to genuine responses; elsewhere, it commits to a dummy symbol $\perp$, with the same message lengths and prescribed reception pattern. Dummy commitments are independent of the challenges and can be prepared in advance. Their transmissions are scheduled to meet the candidate instance’s reception times, whereas genuine commitments depend on challenges received during the physical interaction.

To fix which trajectory subsequent proofs refer to, the prover also sends an extractable commitment $C = \text{Com}(\text{st}; r)$ at initialization. This commitment fixes the claimed trajectory; the commitments $\{c_\alpha\}_{\alpha \in S}$ subsequently supply its physical authentication. Separating these roles prevents the prover from selecting different subtrajectories in different prove phases.¹²

Proof of Consistency. Let τ denote the (classical) transcript of the commit-phase interaction. For a public polynomial-time predicate f , the prover gives a zero-knowledge argument of knowledge for a relation R_f with public input (C, τ, T, f) . The witness consists of a trajectory st , randomness opening C to st , and, for every point α on that trajectory, a response a_α and randomness opening c_α to a_α . The relation checks that:

1. st belongs to the publicly specified family T of admissible trajectories, and the supplied opening of C to st is valid;
2. for every point α on st , the supplied opening of c_α to a_α is valid, and a_α satisfies the underlying position-verification predicate for that instance, using the verifier data and timing information recorded in τ ; and
3. the committed trajectory satisfies $f(\text{st}) = 1$.

Thus, the same witness links the trajectory fixed by C to the responses fixed during the physical interaction and to the requested predicate. Provided that the underlying verification checks are classically computable in polynomial time, R_f is a classical **NP** relation, even if generating valid position-verification responses requires quantum computation. Zero knowledge hides the trajectory, the responses, and their openings.

Extraction with Physical Meaning. The commitment C allows the simulator to extract the a trajectory st^* . Extraction from the response commitments fixes the corresponding responses. Binding and the prove-phase consistency argument ensure that any accepting proof refers to these same values. The remaining task is to connect these extracted strings to physical reality.

Toward that, we use the underlying Π_{PV} as the auxiliary verification protocol. In the security reduction, the simulator interacts with an external position verifier and forwards its fresh challenge to the malicious prover as the challenge of the corresponding commitment instance. When the prover returns c_α , the simulator extracts a_α and forwards it to the external verifier. That is, the reduction places the external challenges inside the adversary’s actual interaction and converts committed responses into ordinary responses online. The consistency relation transfers acceptance to the underlying verification predicate, while position-verification soundness supplies the physical interpretation.

¹² Technically, we also need to run a zero-knowledge argument to prove the consistency of the trajectory commitment C with the response commitments $\{c_\alpha\}_{\alpha \in S}$; this is crucial to make the commit phase itself a well-formed extractable commitment. But we omit this discussion from the overview for simplicity.

Obtaining the UC Realization. We instantiate the commitment and ZKAoK components of this construction with their UC counterparts. This yields a realization of $\mathcal{F}_{\text{CnP}}^{\text{ST}}$ satisfying the UC spatiotemporal 2PC definition from the preceding subsection. In particular, the construction supports concurrent commitment sessions and later proofs about previously authenticated trajectory segments.

Extractable Spatiotemporal Trajectory Commitments. In [Sec. 4.3](#), we also abstract the commitment to trajectories, together with its physical-meaningfulness check, as an *extractable spatiotemporal trajectory commitment*. Its interface exposes a classical commitment record and a classical opening predicate, together with extraction that both preserves a simulated continuation and supports physical certification of the extracted trajectory. We believe that this abstraction provides a simple, modular building block for future applications requiring private, physically authenticated trajectory data.

2.3 From Commit-and-Prove to Spatiotemporal MPC

Our construction starts from a semi-honest MPC protocol and follows the GMW paradigm: each party commits to its inputs and randomness at the outset, and proves that each outgoing message is consistent with these commitments and the prescribed computation [[GMW87](#), [CLOS02](#)]. To evaluate a spatiotemporal functionality, each party additionally commits to a classical string describing its trajectory, which the MPC uses together with the parties’ other inputs. The key additional requirement is that this string must describe the party’s *physical trajectory*, rather than merely a trajectory that it claims to follow.

Ordinary consistency proofs do not establish this physical requirement: a party could execute the prescribed computation correctly using a fabricated trajectory. Our spatiotemporal commit-and-prove protocol supplies the missing authentication. Its commit phase records trajectory information during the execution, and its prove phase establishes consistency between that physically authenticated information and the committed trajectory string. The challenge is to integrate this physical authentication with the computational consistency checks.

Round-by-Round Authentication and Consistency. A naive attempt is to commit to trajectories in parallel with the GMW-style MPC execution; at the end, all parties perform the proof stage of the spatiotemporal commit-and-prove protocol to establish the proofs of physical consistency through the computation. This checks the right condition, but *too late*! Consider a functionality that releases a secret only to a party satisfying a location-dependent condition. Even if every MPC message is consistent with the prescribed computation, a party could supply a fabricated trajectory and obtain the secret before its physical consistency is checked. Rejecting its final proof would not undo that disclosure.

We therefore divide the execution into rounds, with a designated physical observation interval I_r for each round r . During this interval, each party P_b runs the commit phase of our spatiotemporal commit-and-prove protocol for the trajectory segment $\text{st}_{b,r} = \text{st}_b|_{I_r}$, concurrently with the corresponding MPC round. The prove phase then enforces two conditions jointly:

- *Computational consistency*: the party’s round- r messages follow the semi-honest MPC protocol, using its committed inputs, randomness, and trajectory string, together with the preceding accepted transcript.
- *Physical consistency*: the trajectory segment authenticated during I_r agrees with the corresponding segment of the trajectory string committed to at the outset.

In this way, the proof stage both enforces honest behavior in the underlying semi-honest MPC execution and certifies the physical information on which that behavior depends.

The protocol advances to the next round only after all required proofs for the current round have been accepted. Until then, honest parties treat the round’s incoming messages as pending and take no dependent computational step. Crucially, merely delaying the processing of incoming messages is not sufficient: any outgoing message or output whose disclosure relies on the current trajectory segment must itself be withheld until the required physical consistency proofs have been accepted. The final round is subject to the same rule. This ordering prevents an unauthenticated trajectory segment from authorizing a disclosure that a later rejection cannot undo. This yields [Thm. 2](#).

2.4 Lifting the Construction to Quantum Functionalities

We finally allow the parties’ computational inputs and outputs to be quantum, while retaining classical descriptions of their trajectories. First, notice that the ideas developed in [Sec. 2.3](#) does not directly extend, because they require running a consistency (spatiotemporal) proof to check polynomial-time relations between the committed trajectory segments and the party’s round- r computation for each round r ; this works well for classical computations. However, one round of a *quantum* MPC protocol may involve exchanging registers whose correctness may require asserting that a particular transmitted state was obtained by applying a prescribed quantum operation to another state. Such an assertion is not an ordinary NP statement about a classical transcript. Nor does simply replacing NP proofs with proofs for QMA solve the interface problem: standard QMA statements are classical, even though their witnesses may be quantum. We therefore avoid using spatiotemporal proofs to certify quantum message generation.

A First Route: Authenticate the Classical Encoding of a Quantum Computation. The garbling-based approach of [\[BCKM21a\]](#) suggests a way to separate the classical and quantum tasks. It uses the quantum garbled circuit scheme of [\[BY22\]](#), in which garbling a quantum circuit consists of a *classical* procedure that produces an entirely classical garbled program: the quantum information resides in the encoded inputs, rather than in the garbled program itself. Classical secure computation can therefore generate the garbled program and its classical control information, while quantum input encoding and evaluation take place outside that computation. Authentication and consistency mechanisms are developed (thanks to [\[BCKM21a\]](#)) to link these components.

For our purposes, the parties could supply their trajectories as additional private inputs to this classical computation, which programs the target quantum functionality using those values. To avoid revealing a trajectory through the circuit description, one can express the target using a fixed public universal circuit with the trajectories supplied as private classical data.

We then apply a round-by-round compilation similar to [Sec. 2.3](#): recall that the protocol of [\[BCKM21a\]](#) first runs a post-quantum classical MPC to generate the classical garbled circuit and then performs additional quantum rounds for input encoding and evaluation. As explained above, the trajectory values are committed in the classical stage and fixed as private data programming the garbled quantum computation. Using the same spacetime mesh and round-by-round authentication strategy, we run our commit-and-prove protocol in parallel with the entire execution, covering both the classical and quantum stages. At each round boundary, the consistency proofs establish that the trajectory segment authenticated during that round agrees with the corresponding portion of the trajectory already fixed in the garbled computation; no dependent step proceeds until these proofs are accepted. Thus, the checks during the later quantum rounds remain tied to the same trajectory

values “hard-wired” into the (garbled) target quantum circuit, yielding a protocol-specific route to spatiotemporal MPQC.

Although this approach works, it lacks modularity: the consistency argument is tied to the “classical-garbling” computation of the [BCKM21a] construction. Ideally, we would like to have a generic construction that can be applied to any underlying quantum protocol, allowing us to use the underlying quantum protocol as a black box.

A Generic Approach. Our eventual construction requires only a UC-secure MPQC protocol and the spatiotemporal commit-and-prove we build in Sec. 2.2. The central observation is that the interface between these components is entirely classical: both must use the same trajectory.

Each party first commits to its trajectory using an ordinary post-quantum binding commitment, such as Naor’s commitment instantiated with a post-quantum pseudorandom generator. Write $C_b = \text{Com}(\text{st}_b; r_b)$, where C_b denotes the complete public commitment record, including any receiver message required by the scheme. Let $\mathcal{Q}$ be the desired spatiotemporal quantum functionality, and let $\mathbf{C} = \{C_b\}_b$ be the (public) collection of the parties’ trajectory commitments. Instead of evaluating $\mathcal{Q}$ directly, the parties invoke the MPQC protocol for a modified functionality $\mathcal{Q}_{\text{check}}^{\mathbf{C}}$. It receives each party’s quantum input register together with (st_b, r_b) , verifies $\text{Verify}_{\text{Com}}(C_b, \text{st}_b, r_b) = 1$ for each b , and aborts if any check fails. Otherwise, it applies $\mathcal{Q}$ to the joint quantum input and the supplied trajectories. The joint input may be entangled across parties and with an external reference system; no product-state assumption is needed.

In parallel with the MPQC execution, the parties run the commit-and-prove protocol using a similar round-by-round construction as in Sec. 2.3. For each round, the commit phase authenticates the corresponding trajectory segment, and the prove phase establishes that this segment agrees with the corresponding portion of the trajectory committed in C_b . These are the same commitments whose openings are checked by $\mathcal{Q}_{\text{check}}^{\mathbf{C}}$, linking the authenticated segments to the trajectories supplied to the quantum computation. As before, no dependent step proceeds until the required consistency proofs are accepted; in particular, all trajectory segments on which an output depends must be authenticated before that output is released.

The division of responsibility is now explicit. The MPQC protocol guarantees correct quantum computation but does not interpret physical presence. The commit-and-prove protocol certifies physical presence but does not verify quantum computation. The ordinary commitment forces both components to refer to the same classical trajectory.

Returning to Classical Functionalities. The generic construction above treats the underlying MPQC protocol as a black box. In particular, replacing the MPQC protocol with a maliciously secure MPC protocol for classical functionalities yields secure spatiotemporal MPC for classical functionalities. This provides an alternative route to the feasibility results described in Sec. 2.3. In the main body, we present only this generic construction, which handles both classical and quantum functionalities in a unified manner. We nevertheless retain the description in Sec. 2.3 to provide a complementary approach, which may admit/ease further optimizations in future work, such as black-box constructions based on black-box commit-and-prove techniques.

Why an Ordinary Commitment Can Suffice. The linking commitment C_b is not used as a separately UC-realized commitment functionality. In particular, the simulator need not extract a trajectory from C_b alone or open an honest commitment to a newly selected value. Instead, extraction occurs at the stronger interfaces already provided by the construction. In the MPQC hybrid, the simulator obtains the corrupted party’s effective classical trajectory and opening. In the “physical-relevant” hybrid, it obtains the trajectory authenticated by the commit-and-prove

instance. Both must be consistent with the same public record C_b . Statistical binding therefore identifies the two values, while computational hiding protects honest trajectories. This is the reason that the linking commitment need not itself be UC secure.

It is also worth emphasizing that an ordinary commitment may be *malleable* [DDN91], and UC security of the surrounding protocols does not by itself provide non-malleability. This poses no problem for our construction: even if an adversary copies or transforms an honest party's commitment C_b , using the resulting record requires it to provide a valid opening within the quantum computation and to physically certify the same trajectory through its own commit-and-prove interface. The UC simulations for these components capture such behavior even across interleaved sessions with related public inputs, while binding ensures that the computational and physical components cannot accept different trajectories for the same record. A related trajectory that the adversary can both open and physically authenticate is simply an admissible effective input, whereas a copied record without a usable opening is insufficient. Thus, our security argument does not rule out commitment malleation per se; rather, it shows that malleation cannot bypass the interfaces through which the committed value must be supplied and physically certified. These interfaces incorporate session and party identifiers to prevent acceptance in one context from being reused as authorization in another.

The resulting transformation treats the underlying MPQC protocol as a computational component, rather than inspecting its quantum messages or reproducing its internal authentication mechanisms. Under the stated composition and corruption assumptions, it yields [Thm. 3](#).

3 Preliminaries

Basic Notations. Let $\lambda \in \mathbb{N}$ denote security parameter. For a positive integer n , let $[n]$ denote the set $\{1, 2, \dots, n\}$. For a finite set $\mathcal{X}$, $x \leftarrow \mathcal{X}$ means that x is uniformly chosen from $\mathcal{X}$.

A function $f : \mathbb{N} \rightarrow [0, 1]$ is said to be *negligible* if for all polynomial p and sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$; it is said to be *overwhelming* if $1 - f$ is negligible, and said to be *noticeable* if there is a polynomial p such that $f(\lambda) \geq 1/p(\lambda)$ for sufficiently large $\lambda \in \mathbb{N}$. We denote by *poly* an unspecified polynomial and by *negl* an unspecified negligible function.

Unless stated otherwise, all adversaries considered herein are modeled as a non-uniform QPT algorithm, specified by a sequence of polynomial-size quantum circuits with quantum advice $\{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$. In an execution with the security parameter λ , $\mathcal{A}$ runs $\mathcal{A}_\lambda$ taking ρ_λ as the advice. For simplicity, we often omit the index λ and just write $\mathcal{A}(\rho)$ to mean a non-uniform QPT algorithm specified by $\{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$. When it is clear from the context, we also omit the advice ρ and just write $\mathcal{A}$ to mean a non-uniform QPT algorithm.

3.1 Post-Quantum Extractable and Commitment scheme

We define post-quantum extractable commitment schemes with simulation. Our definition is taken almost verbatim from [CCLY22b], with only cosmetic changes to fit our notation.

Definition 2 (Post-Quantum Commitment Scheme [CCLY22b]). A *post-quantum commitment scheme* Π is a protocol between two probabilistic polynomial-time (PPT) machines: a committer $\mathcal{C}$ and a receiver $\mathcal{R}$. Let $m \in \{0, 1\}^{\ell(\lambda)}$ (where λ is the security parameter and $\ell(\cdot)$ is some polynomial) be a message that $\mathcal{C}$ wants to commit to. The protocol consists of the following stages:

- **Commit Stage:** $\mathcal{C}(m)$ and $\mathcal{R}$ interact with each other to generate a commitment com , $\mathcal{C}$'s state ST_C and $\mathcal{R}$'s output state OUT_R . We denote this execution by $(\text{com}, \text{ST}_C, \text{OUT}_R) \leftarrow \langle \mathcal{C}(m), \mathcal{R} \rangle(1^\lambda)$. When we consider a malicious quantum committer $\mathcal{C}^*$, we allow it to generate any quantum state ST_{C^*} . Also, a malicious quantum receiver $\mathcal{R}^*$ can output any quantum state, which we denote by OUT_{R^*} .
- **Decommit Stage:** $\mathcal{C}$ generates a decommitment $\text{decom} \leftarrow \mathcal{S}(\text{ST}_C)$. Then it sends a message m and decommitment decom to $\mathcal{R}$, and $\mathcal{R}$ outputs a bit $b \in \{0, 1\}$ indicating acceptance (i.e., $b = 1$) or rejection (i.e., $b = 0$). We assume that $\mathcal{R}$'s verification procedure is deterministic and denote it by $\text{Verify}(\text{com}, m, \text{decom})$.

The scheme satisfies the following requirement:

- **Correctness.** For any $m \in \{0, 1\}^{\ell(\lambda)}$, it holds that:

$$\Pr \left[\begin{array}{l} (\text{com}, \text{ST}_C, \text{OUT}_R) \leftarrow \langle \mathcal{C}(m), \mathcal{R} \rangle(1^\lambda) \\ b = 1 : \text{decom} \leftarrow \text{ST}_C \\ b \leftarrow \text{Verify}(\text{com}, m, \text{decom}) \end{array} \right] = 1$$

- **Computational Hiding.** For any $m_0, m_1 \in \{0, 1\}^{\ell(\lambda)}$ and any non-uniform QPT receiver $\mathcal{R}^*$, the following holds:

$$\begin{aligned} & \{ \text{OUT}_{R^*} : (\text{com}, \text{ST}_C, \text{OUT}_{R^*}) \leftarrow \langle \mathcal{C}(m_0), \mathcal{R}^* \rangle(1^\lambda) \}_\lambda \\ & \stackrel{c}{\approx} \{ \text{OUT}_{R^*} : (\text{com}, \text{ST}_C, \text{OUT}_{R^*}) \leftarrow \langle \mathcal{C}(m_1), \mathcal{R}^* \rangle(1^\lambda) \}_\lambda. \end{aligned}$$

- **Statistical binding.** For all (potentially unbounded) committer $\mathcal{C}^*$, the following holds:

$$\Pr \left[\begin{array}{l} \exists m_0, m_1, \text{decom}_0, \text{decom}_1, \\ \text{s.t. } m_0 \neq m_1 \wedge \\ \text{Verify}(\text{com}, m_0, \text{decom}_0) = 1 \wedge \\ \text{Verify}(\text{com}, m_1, \text{decom}_1) = 1 \end{array} : (\text{com}, \text{ST}_{C^*}, \text{OUT}_R) \leftarrow \langle \mathcal{C}^*, \mathcal{R} \rangle(1^\lambda) \right] = \text{negl}(\lambda)$$

We define the value function as follows:

Definition 3 (Value Function). For a post-quantum commitment scheme Π satisfying [Def. 4](#), we define the value function as follows:

$$\text{val}_\Pi(\text{com}) := \begin{cases} m & \text{if } \exists \text{ unique } m \text{ s.t. } \exists \text{ decom}, \text{Verify}(\text{com}, m, \text{decom}) = 1 \\ \perp & \text{otherwise} \end{cases}$$

We say that com is valid if $\text{val}_\Pi(\text{com}) \neq \perp$ and invalid if $\text{val}_\Pi(\text{com}) = \perp$.

Then we define the post-quantum extractable commitment with simulation as follows:

Definition 4 (Post-Quantum Extractable Commitment with Simulation). A post-quantum commitment scheme Π is extractable with simulation if for any non-uniform QPT $\mathcal{C}^*$, there exists a QPT algorithm $\mathcal{SE}$ such that

$$\{ \mathcal{SE}(1^\lambda) \}_\lambda \stackrel{c}{\approx} \{ (\text{val}_\Pi(\text{com}), \text{ST}_{C^*}) : (\text{com}, \text{ST}_{C^*}) \leftarrow \langle \mathcal{C}^*, \mathcal{R} \rangle(1^\lambda) \}_\lambda$$

Instantiation. Constructions satisfying [Def. 4](#) are known in the following settings: in the CRS model based on QLWE [\[DFL⁺09\]](#), and in the QROM without any additional assumptions [\[ABKK23\]](#).

3.2 Post-Quantum Zero-Knowledge Arguments

We define post-quantum zero-knowledge arguments.

Definition 5 (Post-Quantum Zero-Knowledge Arguments). Π is an interactive protocol between two probabilistic polynomial-time (PPT) machines: a prover P and a verifier V . After the interaction, V outputs a bit $b \in \{0, 1\}$, where $b = 1$ denotes acceptance and $b = 0$ denotes rejection. We use $b \leftarrow \langle P, V \rangle(1^\lambda, x)$ to denote the verifier's output after the interaction between P and V on common input x and security parameter λ .

We say that Π is a post-quantum zero-knowledge argument for a language $L \in \mathbf{NP}$ if the following properties hold:

1. **Completeness.** For all $x \in L$ and all $w \in \mathcal{R}_L(x)$,

$$\Pr[b = 1 : b \leftarrow \langle P(w), V \rangle(1^\lambda, x)] \geq 1 - \text{negl}(\lambda).$$

2. **Soundness.** For all $x \notin L$ and all non-uniform QPT prover P^* ,

$$\Pr[b = 1 : b \leftarrow \langle P^*, V \rangle(1^\lambda, x)] \leq \text{negl}(\lambda).$$

3. **Zero Knowledge.** For a malicious non-uniform QPT verifier V^* , we denote its interaction with the honest prover as $\text{OUT}_{V^*} \leftarrow \langle P(w), V^* \rangle(1^\lambda, x)$ where OUT_{V^*} is the output of the verifier V^* . There exists a QPT simulator Sim such that for all $x \in L$ and all $w \in \mathcal{R}_L(x)$

$$\{\text{Sim}(1^\lambda, x)\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{\text{OUT}_{V^*} : \text{OUT}_{V^*} \leftarrow \langle P(w), V^* \rangle(1^\lambda, x)\}_{\lambda \in \mathbb{N}}.$$

Instantiation. In the plain model, constructions satisfying Def. 5 are known in polynomial rounds from QLWE [Wat06], and in constant rounds from QLWE together with the existence of quantum fully homomorphic encryption [BS20].

Constant-round constructions are also known in the CRS model under QLWE [DFL⁺09], as well as in the QROM without additional assumptions [ABKK23]. These constructions follow from the observation that post-quantum zero-knowledge arguments can be obtained from post-quantum extractable commitment schemes with only $O(1)$ additional rounds and no additional assumptions, relying on (a straightforward post-quantum extension of) [Ros04]. Moreover, both constructions can be made UC secure if the underlying post-quantum extractable commitment scheme is a UC secure commitment.

3.3 UC Framework

We assume familiarity with the UC framework [Can01]. We provide a brief overview here, following the wording of [GKP17]. A formal treatment can be found in the literature (e.g., [Can01, CLOS02]). Recall that in the UC framework, the model for protocol execution consists of the environment $\mathcal{Z}$, the adversary $\mathcal{A}$, and the parties running protocol π . In this paper, we consider static adversaries and assume the existence of authenticated communication channels. Let $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}(\kappa, z)$ denote a random variable representing the output of $\mathcal{Z}$ on security parameter $\kappa \in \mathbb{N}$ and input $z \in \{0, 1\}^*$, with a uniformly chosen random tape. Let $\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}$ denote the ensemble $\{\text{EXEC}_{\pi, \mathcal{A}, \mathcal{Z}}(\kappa, z)\}_{\kappa \in \mathbb{N}, z \in \{0, 1\}^*}$.

The security of a protocol π is defined using the *ideal protocol*. In an execution of the ideal protocol, all parties simply provide their inputs to the *ideal functionality* $\mathcal{F}$. The ideal functionality $\mathcal{F}$ securely performs the desired task and provides outputs to the parties, which forward these

outputs to $\mathcal{Z}$. The adversary $\mathcal{S}$ in an execution of the ideal protocol is often called the *simulator*. Let $\pi(\mathcal{F})$ denote the ideal protocol for functionality $\mathcal{F}$.

We say that a protocol π *emulates* a protocol ϕ if, for every adversary $\mathcal{A}$, there exists an adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$, on any input, can distinguish with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running π , or with $\mathcal{S}$ and parties running ϕ . We say that π *securely realizes* an ideal functionality $\mathcal{F}$ if it emulates the ideal protocol $\pi(\mathcal{F})$.

Based on the UC framework, we define the UC commitment and UC zero-knowledge proof as follows:

UC Commitment. We define the ideal functionality of $\mathcal{F}_{\text{Com}}$ as follows:

Figure 1: Ideal Functionality $\mathcal{F}_{\text{Com}}$
$\mathcal{F}_{\text{Com}}$ proceeds as follows, running with parties $P_1, \dots, P_n$ and adversary $\mathcal{S}$. – Commit phase. Upon receiving a value $(\text{Commit}, \text{sid}, P_i, P_j, x)$ from P_i, where $x \in \{0, 1\}^m$, record the value x and send the message $(\text{Receipt}, \text{sid}, P_i, P_j, x)$ to P_j. – Open phase. Upon receiving a message $(\text{open}, \text{sid}, P_i, P_j, x)$ from P_i, proceed as follows: If some value x was previously recorded, then send the message $(\text{Open}, \text{sid}, P_i, P_j, x)$ to P_j and $\mathcal{S}$ and halt. Otherwise, halt.

Definition 6 (UC Commitment). *A protocol is a universal composable (UC) commitment if it securely realizes functionality $\mathcal{F}_{\text{Com}}$.*

UC Zero-Knowledge. We define the ideal functionality of $\mathcal{F}_{\text{ZK}}$ as follows:

Figure 2: Ideal Functionality $\mathcal{F}_{\text{ZK}}$
$\mathcal{F}_{\text{ZK}}$ proceeds as follows, running with parties $P_1, \dots, P_n$ and adversary $\mathcal{S}$. The functionality is parameterized by a binary relation R . 1. Wait to receive a value $(\text{verifier}, \text{id}, P_i, P_j, x)$ from some party P_i. Once such a value is received, send $(\text{verifier}, \text{id}, P_i, P_j, x)$ to $\mathcal{S}$ and ignore all subsequent $(\text{verifier}, \cdot)$ values. 2. Upon receipt a value $(\text{prover}, \text{id}, P_i, P_j, x', w)$ from P_j, let $v = 1$ if $x = x'$ and $R(x, w) = 1$ holds, and $v = 0$ otherwise. Send (id, v) to P_i and $\mathcal{S}$ and halt.

Definition 7 (UC Zero-Knowledge). *A protocol is a universal composable (UC) zero-knowledge proof if it securely realizes functionality $\mathcal{F}_{\text{ZK}}$.*

4 Spatiotemporal Verification and Commitment

In this section, we formally introduce the model of spacetime and spacetime trajectory. Starting with a “nice” position verification, we give the definition and protocol of spatiotemporal trajectory commitment.

4.1 The Model

Our model is based on that specified in [LLQ22]. The model consists of three types of parties: provers, verifiers, and adversaries.

- Space and time are continuous, and the clocks of all parties are synchronized. A spacetime point is represented by $\alpha = (L, t) \in \mathbb{R}^d \times \mathbb{R}$, where $L \in \mathbb{R}^d$ denotes the spatial location, d is the spatial dimension, and $t \in \mathbb{R}$ denotes time. In this work, whenever we consider position-verification protocols with privacy guarantees against *malicious* parties, we restrict our attention to the one-dimensional setting, i.e., $d = 1$. See [Sec. 4.3.2](#) for further discussion.
- Each party’s location at a particular time is described by a spacetime point as $\alpha = (L, t) \in \mathbb{R}^d \times \mathbb{R}$.
- Adversaries may also communicate over private quantum channels, thereby preventing the verifiers from detecting malicious activity through these communications.
- The adversary considered herein may consist of multiple parties. For a coalition of quantum adversaries, we distinguish between the following two settings:
 - **Unentangled adversaries:** The adversarial parties share no entanglement before the protocol begins.
 - **Entangled adversaries:** The adversarial parties may share an arbitrary polynomial amount of entanglement before the protocol begins.

However, explicitly accounting for both types of adversaries throughout all our definitions would be cumbersome. We therefore use the phrase “a coalition of non-uniform QPT adversaries” in our definitions (e.g., in [Def. 12](#)) as shorthand for two separate security notions: one against unentangled adversaries and the other against entangled adversaries. When presenting our constructions, security theorems, and proofs, we will then treat the two settings separately and explicitly state the hardness and model assumptions required for each.

- All computations are performed instantaneously, while messages transmitted over any communication channel propagate at unit speed (the speed of light).

We next introduce the notion of a spatiotemporal trajectory. In what follows, most definitions and constructions are formulated in terms of a party’s trajectory rather than a single spacetime point.

Definition 8 (Spatiotemporal Trajectory). *A spatiotemporal trajectory over $\mathbb{R}^d \times \mathbb{R}$ is an ordered set*

$$(\mathbf{st}, \prec_{\text{time}}),$$

where, for some positive integer m ,

$$\mathbf{st} = \{(L_i, T_i) : i \in [m]\} \subseteq \mathbb{R}^d \times \mathbb{R}$$

is a set of m observations with pairwise distinct timestamps, i.e.,

$$i \neq j \implies T_i \neq T_j,$$

and $\prec_{\text{time}}$ is the strict total order induced by the timestamps:

$$(L_i, T_i) \prec_{\text{time}} (L_j, T_j) \iff T_i < T_j.$$

Throughout this paper, a spatiotemporal trajectory is always ordered by its timestamps. We therefore simply write $\mathbf{st}$ for the ordered set $(\mathbf{st}, \prec_{\text{time}})$. By a slight abuse of notation, when S is a possibly unordered set of spatiotemporal points, we write

$$\mathbf{st} \subseteq S$$

to mean that the underlying set $\mathbf{st}$ satisfies $\mathbf{st} \subseteq S$, disregarding the order. The intended interpretation will be clear from context.

4.2 Verification of Position and Spatiotemporal Trajectory

We first present the definition of position verification, following the formalism in [GGG⁺26].

Definition 9 (Position Verification). Let $\Pi = \{\mathcal{P}, \mathcal{V}\}$ be an interactive protocol consisting of two QPT machines $\mathcal{P}$ and $\mathcal{V}$. The scheme should satisfy the following properties:

- **Completeness:** $\mathcal{P}$ and $\mathcal{V}$ hold the security parameter 1^λ and a spacetime point α as their public input. After the interaction, $\mathcal{V}$ outputs a bit $b \in \{0, 1\}$, where $b = 1$ denotes acceptance and $b = 0$ denotes rejection. We denote the interaction by $b \leftarrow \langle \mathcal{P}, \mathcal{V} \rangle(1^\lambda, \alpha)$. It holds that

$$\Pr[b = 1 : b \leftarrow \langle \mathcal{P}, \mathcal{V} \rangle(1^\lambda, \alpha)] = 1 - \text{negl}(\lambda).$$

- **Soundness:** For (a collection of) malicious QPT provers $\mathcal{P}^*$, let $\mathbf{R}^* = (\alpha_1^*, \dots, \alpha_k^*)$ denote the set of spacetime points that $\mathcal{P}^*$ occupies. Let $b \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle(1^\lambda, \alpha)$ denote the interaction between $\mathcal{P}^*$ and $\mathcal{V}$, where b is the verifier's output. For all $\alpha \notin \mathbf{R}^*$, it holds that

$$\Pr[b = 1 : b \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle(1^\lambda, \alpha)] = \text{negl}(\lambda).$$

One may further require that the position-verification protocol satisfy the following “nice” property introduced in [GGG⁺26].

Definition 10 (Nice Position Verification Protocol). We say a position verification protocol Π (as per Def. 9) is nice if it satisfies the following properties.

- Π is a two-message, one-round challenge–response protocol in which the verifiers send challenges and the prover returns a response. For each position α , the verifiers apply the predicate $W_\alpha(\cdot)$ to the prover's response to verify the claimed position.
- The predicate $W_\alpha(\cdot)$ used to verify the position α by $\mathcal{V}$ is deterministic and classical.
- For all $\alpha = (L, t) \in \mathbb{R}^d \times \mathbb{R}$, there exist static verifier locations $X_1^S, \dots, X_k^S \in \mathbb{R}^d$ whose convex hull contains the spatial point L .

Remark 1. A position-verification protocol directly verifies only a spatial location, rather than a spatiotemporal point. However, the location can be associated with the time at which the protocol is executed. Thus, without loss of generality, we treat α as a predicate that verifies a spatiotemporal point.

Then, based on the notion of spatiotemporal trajectory Def. 8, we provide the definition of spatiotemporal trajectory verification.

Definition 11 (Spatiotemporal Trajectory Verification (STV)). A spatiotemporal trajectory verification scheme $\Pi = (\mathcal{P}, \mathcal{V}, \mathbf{T})$ consists of two QPT machines $\mathcal{P}$ (dubbed the prover) and $\mathcal{V}$ (dubbed the verifier), and a so-called admissible trajectory set $\mathbf{T}$, which is a finite set where each element $\mathbf{st} \in \mathbf{T}$ is a spatiotemporal trajectory over $\mathbb{R}^d \times \mathbb{R}$ (as per Def. 8). The scheme should satisfy the following properties:

1. **Completeness:** $\mathcal{P}$ and $\mathcal{V}$, holding the security parameter 1^λ and a trajectory $\mathbf{st} \in \mathbf{T}$ as their public input, interact within a time window during which $\mathcal{P}$ moves along the trajectory as specified by $\mathbf{st}$. After the interaction, $\mathcal{V}$ outputs a bit $b \in \{0, 1\}$ indicating acceptance (i.e., $b = 1$) or rejection (i.e., $b = 0$). We denote this execution by $b \leftarrow \langle \mathcal{P}, \mathcal{V} \rangle(1^\lambda, \mathbf{st})$.

Using this notation, completeness requires that for all $\mathbf{st} \in \mathbf{T}$, it holds that

$$\Pr[b = 1 : b \leftarrow \langle \mathcal{P}, \mathcal{V} \rangle(1^\lambda, \mathbf{st})] = 1 - \text{negl}(\lambda).$$

2. **Soundness:** For a (coalition of) malicious $\mathcal{P}^*$ interacting with the honest $\mathcal{V}$, let $S^* = \{(L_i, T_i)\}_i$ denote the set of all spatiotemporal points that (some part of) $\mathcal{P}^*$ has occupied (see the discussion in [Rmk. 2](#)) during the interaction. We denote this execution as $b \leftarrow \langle \mathcal{P}^*[S^*], \mathcal{V} \rangle(1^\lambda, \text{st})$, where b is the output of $\mathcal{V}$ at the end of the interaction.

Using this notation, soundness requires that for all $\text{st} \in \mathbb{T}$ and all a (coalition of) non-uniform QPT $\mathcal{P}^*$ and its associated S^* , it holds that

$$\Pr \left[(b = 1) \wedge (\text{st} \notin S^*) : b \leftarrow \langle \mathcal{P}^*[S^*], \mathcal{V} \rangle(1^\lambda, \text{st}) = 1 \right] = \text{negl}(\lambda).$$

This construction builds on the position-verification protocol. For a trajectory st , the verifiers execute the position-verification protocol to verify each spacetime point $\alpha \in \text{st}$.

4.3 Spatiotemporal Trajectory Commitment

In this part, we formally define the spatiotemporal trajectory commitment and present a construction.

4.3.1 Definition

In this section, we formally define spatiotemporal trajectory commitment schemes. We consider two variants: a standard scheme [Def. 12](#) and an extractable scheme [Def. 14](#). The latter strengthens the former by additionally guaranteeing that the committed trajectory can be efficiently extracted without affecting the adversary's view.

Definition 12 (Spatiotemporal Trajectory Commitment). A spatiotemporal trajectory commitment scheme $\Pi = (\mathcal{C}, \mathcal{R}, \mathbb{T})$ consists of two QPT machines $\mathcal{C}$ (dubbed the committer) and $\mathcal{R}$ (dubbed the receiver), and a so-called committable-trajectory set $\mathbb{T}$, which is a finite set where each element $\text{st} \in \mathbb{T}$ is a spatiotemporal trajectory over $\mathbb{R}^d \times \mathbb{R}$ (as per [Def. 8](#)). The scheme satisfies the following requirements:

The interaction between $\mathcal{C}$ and $\mathcal{R}$ is split into two stages, **Commit** and **Decommit**, which have the following syntax:

- **Commit:** $\mathcal{C}$ and $\mathcal{R}$, holding the security parameter 1^λ as their common input, interact within a time window during which $\mathcal{C}$ moves along a trajectory $\text{st} \in \mathbb{T}$. After the interaction, the $\mathcal{C}$ outputs a state $\rho_{\mathcal{C}}$ and $\mathcal{R}$ a state $\rho_{\mathcal{R}}$, which will be used in the later **Decommit** stage; $\mathcal{R}$ also outputs a bit b_{com} indicating acceptance (i.e., $b_{\text{com}} = 1$) or rejection (i.e., $b_{\text{com}} = 0$) of the commit-stage interaction; and let τ denote the collection of the classical parts across all the messages exchanged between $\mathcal{C}$ and $\mathcal{R}$ during the interaction. We denote this execution as¹³

$$(\rho_{\mathcal{C}}, \tau, \rho_{\mathcal{R}}, b_{\text{com}}) \leftarrow \langle \mathcal{C}_{\text{st}}, \mathcal{R} \rangle(1^\lambda).$$

- **Decommit:** $\mathcal{C}$ on input $\rho_{\mathcal{C}}$ generates the decommitment information decom and sends $(\text{st}, \text{decom})$ to $\mathcal{R}$. Then, $\mathcal{R}$ computes a (possibly quantum) predicate Verify , which on input st , decom , $\rho_{\mathcal{R}}$, and τ outputs a bit b_{dec} indicating acceptance (i.e., $b_{\text{dec}} = 1$) or rejection (i.e., $b_{\text{dec}} = 0$) of the decommitment.¹⁴ We denote this execution as

$$(\text{st}, \text{decom}) \leftarrow \mathcal{C}(\rho_{\mathcal{C}}); b_{\text{dec}} \leftarrow \text{Verify}(\text{st}, \text{decom}, \rho_{\mathcal{R}}, \tau).$$

¹³ Technically, the classical part τ of the message exchanged during **Commit**, as well as the bit b_{com} , could be included in $\rho_{\mathcal{R}}$. However, we treat them as a separate output, as this convention will be useful for subsequent definitions, e.g., [Def. 14](#).

¹⁴ W.l.o.g., we take the convention that $\text{Verify}(\cdot, \cdot, \cdot, \cdot)$ always outputs 0 whenever $b_{\text{com}} = 0$.

The scheme should satisfy completeness, computational hiding, and **either** statistical binding **or** computational binding, as defined below:

1. **Correctness:** For all $\text{st} \in \mathbb{T}$, it holds that

$$\Pr \left[\begin{array}{l} (\rho_{\mathcal{C}}, \tau, \rho_{\mathcal{R}}, b_{\text{com}}) \leftarrow \langle \mathcal{C}_{\text{st}}, \mathcal{R} \rangle(1^\lambda); \\ b_{\text{com}} = b_{\text{dec}} = 1 : (\text{st}, \text{decom}) \leftarrow \mathcal{C}(\rho_{\mathcal{C}}); \\ b_{\text{dec}} \leftarrow \text{Verify}(\text{st}, \text{decom}, \rho_{\mathcal{R}}, \tau) \end{array} \right] = 1 - \text{negl}(\lambda).$$

2. **Computational Hiding:** For a malicious receiver $\mathcal{R}^*$ and an honest $\mathcal{C}$ moving along a trajectory st , we denote the commit-stage interaction as $\rho_{\mathcal{R}^*} \leftarrow \langle \mathcal{C}_{\text{st}}, \mathcal{R}^* \rangle(1^\lambda)$, where $\rho_{\mathcal{R}^*}$ denotes the commit-stage output of $\mathcal{R}^*$.

Using this notation, computational hiding requires that for all QPT $\mathcal{R}^*$, there exists a QPT machine Sim (dubbed the simulator) such that for all committable trajectories $\text{st} \in \mathbb{T}$, it holds that

$$\{\text{Sim}(1^\lambda)\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{\rho_{\mathcal{R}^*} : \rho_{\mathcal{R}^*} \leftarrow \langle \mathcal{C}_{\text{st}}, \mathcal{R}^* \rangle(1^\lambda)\}_{\lambda \in \mathbb{N}}.$$

3. **Computational Binding:** For a (coalition of) quantum $\mathcal{C}^*$ interacting with the honest $\mathcal{R}$, let $\mathbf{S}^* = \{(L_i, T_i)\}_i$ denote the set of spatiotemporal points that (some part of) $\mathcal{C}^*$ has occupied (see the discussion in [Rmk. 2](#)) during the commit-stage interaction. We denote the commit-stage execution as $(\tau, \rho_{\mathcal{R}}, b_{\text{com}}) \leftarrow \langle \mathcal{C}^*[\mathbf{S}^*], \mathcal{R} \rangle(1^\lambda)$, where τ denotes the collection of the classical parts across all the messages exchanged between $\mathcal{C}^*[\mathbf{S}^*]$ and $\mathcal{R}$ during the commit-stage, $\rho_{\mathcal{R}}$ denotes $\mathcal{R}$'s commit-stage output, and b_{com} is $\mathcal{R}$'s bit indicating acceptance (i.e., $b_{\text{com}} = 1$) or rejection (i.e., $b_{\text{com}} = 0$) of the commit-stage interaction.

Using this notation, computational binding requires that for any (coalition of) non-uniform QPT $\mathcal{C}^*$ and her associated $\mathbf{S}^*$, with probability at least $1 - \text{negl}(\lambda)$ over the commit-stage execution $(\tau, \rho_{\mathcal{R}}, b_{\text{com}}) \leftarrow \langle \mathcal{C}^*[\mathbf{S}^*], \mathcal{R} \rangle(1^\lambda)$, there exists a unique $\text{st}^* \in \mathbb{T}$ such that the following two properties hold:

- (a) $\text{st}^* \subseteq \mathbf{S}^*$, and
- (b) for all other trajectories $\text{st} \neq \text{st}^*$, it holds that

$$\Pr[\exists \text{decom s.t. } \text{Verify}(\text{st}, \text{decom}, \rho_{\mathcal{R}}, \tau) = 1] = \text{negl}(\lambda).$$

Remark 2 (On Malicious Committer's Trajectory). Unlike for an honest committer, the notion of a trajectory for a malicious committer $\mathcal{C}^*$ is subtle to define. This is because $\mathcal{C}^*$ may consist of a collection of QPT machines occupying multiple spatial locations simultaneously. For example, $\mathcal{C}^*$ may comprise three parties $\mathcal{C}_1^*, \mathcal{C}_2^*, \mathcal{C}_3^*$ that remain at three distinct locations L_1, L_2, L_3 , respectively, throughout the commit stage. These parties could then easily emulate a single moving party following the trajectory

$$\text{st} = \{(L_1, T_1), (L_2, T_2), (L_3, T_3)\}$$

for any time points $T_1 < T_2 < T_3$.

We emphasize that [Def. 12](#) is **not** intended to distinguish between these two scenarios. In particular, a distributed $\mathcal{C}^*$ that emulates a single party moving along a trajectory st is not considered to violate the binding guarantee. Accordingly, rather than specifying a single trajectory, we specify the set of spacetime points $\mathbf{S}^*$ occupied by $\mathcal{C}^*$ during the commit stage. Any subset $\text{st} \subseteq \mathbf{S}^*$ is then regarded as a valid trajectory that $\mathcal{C}^*$ may claim to have traversed during that stage.

As mentioned earlier, the recent work by Bartusek et al. [[BHOY26](#)] introduced a notion called *quantum localization* to capture precisely the distinction between a single moving party and a distributed party. However, [Def. 12](#), as well as the other definitions in this paper, does not make this distinction. We leave exploring the connection between our notion of spatiotemporal trajectory commitment and quantum localization as an open question, as discussed previously in [Sec. 1.2](#).

Before presenting the extractable variant of spatiotemporal trajectory commitment, we first introduce the notion of a “committed value.” It will be used in [Def. 14](#) to avoid the possibility that the extracted value is not unique.

Definition 13 (Committed Value). For a spatiotemporal trajectory commitment scheme $\Pi = (\mathcal{C}, \mathcal{R}, \mathsf{T})$ (satisfying [Def. 12](#)), consider the commit-stage execution $(\tau, \rho_{\mathcal{R}}, b_{\text{com}}) \leftarrow \langle \mathcal{C}^*[\mathsf{S}^*], \mathcal{R} \rangle(1^\lambda)$ with a (potentially malicious) committer $\mathcal{C}^*[\mathsf{S}^*]$, where we use the same notation as in the computational binding party of [Def. 12](#). We defined the committed value of such an execution as:

$$\text{val}_\Pi(\tau, \rho_{\mathcal{R}}) := \begin{cases} \text{st} & \text{if } \exists \text{ a unique st} \in \mathsf{T} \text{ s.t. } \exists \text{ decom s.t. } \text{Verify}(\text{st}, \text{decom}, \rho_{\mathcal{R}}, \tau) = 1 \\ \perp & \text{otherwise} \end{cases}.$$

Definition 14 (Extractable Spatiotemporal Trajectory Commitment). A spatiotemporal trajectory commitment scheme $\Pi = (\mathcal{C}, \mathcal{R}, \mathsf{T})$ (satisfying [Def. 12](#)) is said to be extractable if it additionally satisfies the following **Extractability** requirement:

For a (coalition of) quantum $\mathcal{C}^*$ interacting with the honest $\mathcal{R}$, let $\mathsf{S}^* = \{(L_i, T_i)\}_i$ denote the set of spatiotemporal points that (some part of) $\mathcal{C}^*$ has occupied (see the discussion in [Rmk. 2](#)) during the commit-stage interaction. We denote the commit-stage execution as

$$(\mathsf{ST}_{\mathcal{C}^*}, \tau, \rho_{\mathcal{R}}, b_{\text{com}}) \leftarrow \langle \mathcal{C}^*[\mathsf{S}^*], \mathcal{R} \rangle(1^\lambda),$$

where τ denotes the collection of the classical parts across all the messages exchanged between $\mathcal{C}^*[\mathsf{S}^*]$ and $\mathcal{R}$ during the commit-stage, $\mathsf{ST}_{\mathcal{C}^*}$ denotes $\mathcal{C}^*[\mathsf{S}^*]$ ’s commit-stage output, $\rho_{\mathcal{R}}$ denotes $\mathcal{R}$ ’s commit-stage output, and b_{com} is $\mathcal{R}$ ’s bit indicating acceptance (i.e., $b_{\text{com}} = 1$) or rejection (i.e., $b_{\text{com}} = 0$) of the commit-stage interaction.

Using this notation, we require that for any (coalition of) non-uniform QPT $\mathcal{C}^*$ and her associated S^* , there exists a two-stage extractor $\text{Ext} = (\text{Ext}_0, \text{Ext}_1)$, a spatiotemporal verification protocol $\Pi' = (\mathcal{P}', \mathcal{V}', \mathsf{T}')$ (as per [Def. 11](#)), and a polynomial $\text{poly}(\cdot)$ such that the following holds:

- No parts of $\text{Ext} = (\text{Ext}_0, \text{Ext}_1)$ can occupy any spatiotemporal point outside the set S^* ;
- Ext_0 takes as input the security parameter 1^λ and outputs a pair (st^*, ρ) ;
- Ext_1 has two modes of operation, indicated by a command string $\text{cmd} \in \{\text{sim}, \text{prv}\}$. When invoked on input $(\text{sim}, 1^\lambda, \text{st}^*, \rho)$, it outputs a tuple $(\widetilde{\mathsf{ST}}_{\mathcal{C}^*}, \widetilde{b}_{\text{com}})$; when invoked on input $(\text{prv}, 1^\lambda, \text{st}^*, \rho)$, it could act as a prover strategy for the spatiotemporal verification protocol Π' .

These algorithms further satisfy the following properties:

1. **Simulation:** it holds that

$$\begin{aligned} & \left\{ (\Gamma_{\widetilde{b}_{\text{com}}}(\text{st}^*), \widetilde{\mathsf{ST}}_{\mathcal{C}^*}) : \begin{array}{l} (\text{st}^*, \rho) \leftarrow \text{Ext}_0(1^\lambda) \\ (\widetilde{\mathsf{ST}}_{\mathcal{C}^*}, \widetilde{b}_{\text{com}}) \leftarrow \text{Ext}_1(\text{sim}, 1^\lambda, \text{st}^*, \rho) \end{array} \right\}_{\lambda \in \mathbb{N}} \\ & \stackrel{c}{\approx} \left\{ (\text{val}_\Pi(\tau, \rho_{\mathcal{R}}), \mathsf{ST}_{\mathcal{C}^*}) : (\mathsf{ST}_{\mathcal{C}^*}, \tau, \rho_{\mathcal{R}}, b_{\text{com}}) \leftarrow \langle \mathcal{C}^*[\mathsf{S}^*], \mathcal{R} \rangle(1^\lambda) \right\}_{\lambda \in \mathbb{N}}, \end{aligned}$$

where $\Gamma_{\widetilde{b}_{\text{com}}}(\text{st}^*)$ is defined as follows: $\Gamma_{\widetilde{b}_{\text{com}}}(\text{st}^*) := \begin{cases} \text{st}^* & \widetilde{b}_{\text{com}} = 1 \\ \perp & \widetilde{b}_{\text{com}} = 0 \end{cases}$.

2. **Spatiotemporal Extraction:** If

$$\delta(\lambda) := \Pr[b_{\text{com}} = 1 : (\mathsf{ST}_{\mathcal{C}^*}, \tau, \rho_{\mathcal{R}}, b_{\text{com}}) \leftarrow \langle \mathcal{C}^*[\mathsf{S}^*], \mathcal{R} \rangle(1^\lambda)] \geq \kappa(\lambda)^{15},$$

¹⁵ All constructions in this work achieve security with respect to a negligible knowledge-error function $\kappa(\lambda)$, which is left implicit throughout.

then

$$\Pr \left[(\text{st}^* \in \mathcal{T}) \wedge (b' = 1) : \begin{array}{l} (\text{st}^*, \rho) \leftarrow \text{Ext}_0(1^\lambda); \\ b' \leftarrow \langle \text{Ext}_1(\text{prv}, \rho), \mathcal{V}' \rangle(1^\lambda, \text{st}^*) \end{array} \right] = \text{poly}(\delta(\lambda)),$$

where $b' \leftarrow \langle \text{Ext}_1(\text{prv}, \rho), \mathcal{V}' \rangle(1^\lambda, \text{st}^*)$ denotes the execution of Π' between $\text{Ext}_1(\text{prv}, 1^\lambda, \text{st}^*, \rho)$ acting as the prover and the honest verifier $\mathcal{V}'(1^\lambda, \text{st}^*)$, who outputs b' after the interaction.

Remark 3 (Comparison with Prior Work). Compared with the notion introduced in [GGG⁺26], our notion provides two main extensions. First, it considers an entire spatiotemporal trajectory rather than a single spacetime point. This enables the verification of statements about movement over time—for example, that Alice remained within a specified region throughout a given time interval—rather than merely statements asserting her presence at a particular location and time. Second, our notion provides an extractability property by simulation-while-extracting. This property makes the notion more general and facilitates its application to secure multiparty computation.

4.3.2 Spatial Dimension as a Barrier to Privacy

Our treatment of privacy must account for a physical obstacle identified by Girish et al. [GGG⁺26, Section 5]. In a model allowing unrestricted directional communication, a malicious verifier can selectively deliver protocol messages along a chosen ray, or withhold them from selected directions, and observe whether the interaction succeeds. Whenever successful participation depends on receiving these messages, the resulting behavior can reveal whether the prover lies in the targeted region. The same obstacle applies to our work: a malicious receiver may use selective delivery to learn geometric information about the committer’s trajectory, even if the cryptographic contents of the commitment are hiding. For example, successful participation following a ray-restricted challenge can reveal that the committer’s position at the relevant time lies on that ray. This obstacle is inherent to the unrestricted directional-communication model, rather than specific to our definition or construction.

Scope of this work. Accordingly, throughout this paper we consider a one-dimensional spatial interval, with the receiver’s two signal towers fixed at its endpoints and the committer’s admissible trajectory contained within the interval. This geometry eliminates the directional-message attack: from either tower, every admissible interior position lies on the same inward-pointing ray. Consequently, a malicious receiver cannot use the choice of transmission direction to selectively target some interior positions while excluding others. The directional attack therefore reveals no additional spatial information beyond the publicly specified restriction that the committer lies within the interval. Fixing the towers’ locations does not require the receiver to behave honestly; our security guarantees allow malicious behavior within this physical model. All our constructions, including trajectory commitments, commit-and-prove protocols, and the spatiotemporal MPC protocols developed in subsequent sections, are formulated in this one-dimensional setting.

Relation to higher-dimensional extensions. It is worth noting that our constructions and techniques are “orthogonal” to the directional-delivery obstacle in a modular sense: they rely on the interfaces and security guarantees of the underlying spatiotemporal primitives, rather than on a particular mechanism for preventing directional leakage. Girish et al. [GGG⁺26, Section 5] suggest some potential approaches to overcoming this obstacle, including restricting verifiers to broadcast transmissions, enabling provers to distinguish broadcast from directional signals, and using redundant verifier networks under an honest-majority assumption. If a higher-dimensional model or technique (e.g., [GGG⁺26, Section 5]) resolves this obstacle and provides primitives satisfying the

same interfaces and security requirements used here, our constructions and composition arguments carry over to that setting.

4.3.3 Construction

In this section, we present a construction of an extractable spatiotemporal trajectory commitment scheme. Our construction extends the idea underlying [GGG⁺26] from a single spacetime point to an entire spatiotemporal trajectory.

We use the following building blocks:

- An extractable post-quantum commitment scheme $\text{ExtCom} = \langle C, R \rangle$ satisfying Def. 4; let r denote the number of rounds for this protocol.
- A “nice” position-verification protocol Π_{PV} satisfying Def. 10. Note that this protocol has two rounds, For each spacetime point α , let W_α denote the verification predicate associated with the corresponding position-verification protocol Π_α .
- A post-quantum zero-knowledge proof system Π_{ZK} satisfying Def. 5.

We construct an extractable spatiotemporal trajectory commitment scheme $\Pi = (C, R, \mathsf{T})$ as follows. In the one-dimensional setting, it suffices to employ two coordinators positioned at distinct locations that delimit the relevant spatial interval.

Protocol 1: Extractable Spatiotemporal Trajectory Commitment

Setup: The protocol uses the following setup:

- The protocol is parameterized by a security parameter 1^λ and a nonempty set T of admissible spatiotemporal trajectories.
- A collection of designated $R = (R_1, R_2)$ is chosen as the coordinator.
- Let Δ denote the maximum time required for a message to travel from any admissible space point^a to any receiver R_i . Let Δ' denote the minimal time required for a message to travel from any admissible point to any receiver R_i .
- The protocol will execute within a time interval $[t_{\text{init}}, t_{\text{final}}]$, which is chosen such that $t_{\text{init}} < t_{\min}(\mathsf{T}) \leq t_{\max}(\mathsf{T}) < t_{\text{final}}$. More accurately, $t_{\text{init}} := t_{\min}(\mathsf{T}) - (r+1)\Delta$ and $t_{\text{final}} := t_{\max}(\mathsf{T}) + \Delta$.
- Let L denote the set of all location L_i ’s in all trajectories in T . Without loss of generality (e.g., by proper rounding to nearest space point), we assume that $|\mathsf{L}|$ is a polynomial of λ . That is, the supporting space of $\mathcal{T}$ is meshed into polynomially many points.
- The time interval $[t_{\min}(\mathsf{T}) - \Delta, t_{\max}(\mathsf{T})]$ is also meshed into a polynomial-size sequence of time steps $(t_1, t_2, \dots, t_n)$ of equal length, where $t_1 = t_{\min}(\mathsf{T}) - \Delta$ and $t_n = t_{\max}(\mathsf{T})$, where n is a polynomial of λ .

Private Input: The committer C follows a trajectory $\text{st} = \{(L_1, T_1), (L_2, T_2), \dots, (L_m, T_m)\} \in \mathsf{T}$, which it holds as its private input and to which it commits.

Commit Phase.

1. At time t_{init} , C samples one-time-pad keys $\{\text{sk}_{v,u}\}_{v \in [n], u \in [|\mathsf{L}|]}$, where each $\text{sk}_{v,u} \leftarrow \{0, 1\}^{\ell(\lambda)}$ is a uniformly random string of length $\ell(\lambda)$, and $\ell(\lambda)$ denote the length of the prover’s message in the position-verification protocol Π_{PV} . C then commits to st and $\{\text{sk}_{v,u}\}_{v \in [n], u \in [|\mathsf{L}|]}$ using ExtCom , we denote the commitment transcript as com .

2. For each time step t_v in the sequence $(t_1, t_2, \dots, t_n)$ and $t_v \leq t_{\max}(\mathsf{T}) - \Delta'$, R sends the first-round message (i.e., the challenge) of a fresh instance of Π_{PV} . We denote the instance of Π_{PV} initiated at time t_v as $\Pi_{PV}^{(t_v)}$. The challenge is broadcast to all locations in L .
3. C then behaves as follows. For each time point t_v and each space point $L_u \in \mathsf{L}$, let $y_{v,u}$ denote the response to the $\Pi_{PV}^{(t_v)}$ instance **if C were to be at location L_u** . C sends the following $\{c_{v,u}\}_{v \in [n], u \in [\mathsf{L}]}$ to R with the correct timing:

$$c_{v,u} := \begin{cases} \mathsf{sk}_{v,u} \oplus y_{v,u} & \text{if there exists a } T \text{ such that } (L_u, T) \in \mathsf{st} \\ \mathsf{sk}_{v,u} \oplus \perp & \text{otherwise} \end{cases},$$

where $\perp$ can be understood as a publicly designated dummy string (e.g., all zeros) representing an invalid or unsuccessful response.

By “correct timing,” we mean that the “true” response $\mathsf{sk}_{v,u} \oplus y_{v,u}$ is sent right at the time that the $\Pi_{PV}^{(t_v)}$ challenge arrives at C ’s real location L_u , while the “dummy” response $\mathsf{sk}_{v,u} \oplus \perp$ is sent at a time that consists of the $\Pi_{PV}^{(t_v)}$ timing constraints **if C were to be at location L_u** . Note that our construction allows this because t_1 is set to $t_{\min}(\mathsf{T}) - \Delta$, and thus C can plan ahead of time to send the dummy responses at the right time.

4. (**Consistency Proof.**) At the time t_{final} , let M denote combined transcript exchanged between C and R during the time interval $[t_{\text{init}}, t_{\text{final}}]$.

Starting from time t_{final} , C and R involve in the execution of the zero-knowledge argument where C proves honest behavior as described above. More precisely, the common input is $x := (\mathsf{M}, \text{com})$, and the witness is $w := (\{\mathsf{sk}_{v,u}\}, \mathsf{st}, \text{decom})$; C proves that (x, w) satisfies the following conditions:

- $\mathsf{st} \in \mathsf{T}$;
- Using decom as the decommitment information, com is indeed a valid commitment to strings st and $\{\mathsf{sk}_{v,u}\}_{v \in [n], u \in [\mathsf{L}]}$;
- For each $c_{v,u}$ (where u determines a unique $L_u \in \mathsf{L}$) in $\{c_{v,u}\}_{v \in [n], u \in [\mathsf{L}]}$
 - if there exists some (unique, by [Def. 8](#)) T such that $(L_u, T) \in \mathsf{st}$, then $c_{v,u}$ is equal to $\mathsf{sk}_{v,u} \oplus y_{v,u}$ where $y_{v,u}$ is a correct response to the $\Pi_{PV}^{(t_v)}$ instance, consistent with the constraint that C is located at L_u at time T .
 - if there does not exist any T such that $(L_u, T) \in \mathsf{st}$, then $c_{v,u}$ is equal to $\mathsf{sk}_{v,u} \oplus \perp$.

Importantly, this execution of ZK protocol needs to satisfies the timing constraints that C pretends to be at the farthest location in L . That is, for the k -th prover message of this ZK protocol, C needs to time it such that it arrives at R at time $t_{\text{final}} + k\Delta$.

5. R outputs b_{com} that is set to be the output of R acting as the verifier of this ZK protocol.

Decommit Phase.

1. C generates the decommitment information decom and sends $(\{\mathsf{sk}_{v,u}\}_{v \in [n], u \in [\mathsf{L}]}, \mathsf{st}, \text{decom})$ to R .
2. R accepts if the decommitment information is valid; otherwise, it rejects.

^a The set of admissible points consists of all space points appearing in any trajectory $\mathsf{st} \in \mathsf{T}$.

Lemma 1 (Security of the Spatiotemporal Trajectory Commitment Construction).

Assume that Π_{PV} is a nice position-verification protocol satisfying Def. 10 against unentangled (resp., entangled) adversaries. Assume further that Π_{ZK} is a post-quantum zero-knowledge proof system satisfying Def. 5, and ExtCom is a post-quantum extractable commitment scheme satisfying Def. 4. Then, Prot. 1 is a spatiotemporal trajectory commitment scheme satisfying Def. 14 against unentangled (resp., entangled) adversaries.

Proof. Completeness. Consider an honest prover P_{st} following an admissible trajectory $\text{st} \in \mathcal{T}$. At every point $\alpha \in \text{st}$, P_{st} and the verifier execute the corresponding position-verification protocol Π_{α} , with the prover's responses encrypted under the honestly generated secret key. By the completeness of Π_{α} and the correctness of the XOR operation, the encrypted transcript decrypts to responses satisfying the corresponding verification predicate W_{α} , except with negligible probability. Moreover, the correctness of the commitment scheme implies that the honestly generated commitment can be successfully opened. Hence, the statement proved in zero knowledge is true, and the verifier accepts the proof by the completeness of the zero-knowledge proof system, except with negligible probability. It follows that both the commit and decommit phases are accepted. Taking a union bound over all $\alpha \in \text{st}$, the overall failure probability remains negligible.

Computational hiding. Let R^* be any malicious QPT receiver. We construct a QPT simulator $\mathcal{S}$ that emulates the joint view of R^* throughout the commit phase without knowledge of the committed trajectory. The simulator replaces the commitment to $(\{\text{sk}_{v,u}\}_{v \in [n], u \in [|\mathcal{L}|]}, \text{st})$ with a commitment to an appropriately chosen dummy value, replaces all the encrypted position verification responses with encryptions of dummy messages of the same length, and simulates the zero-knowledge proof. By the computational hiding of the commitment scheme ExtCom , the pseudorandom of the XOR operation, and the zero-knowledge property of the proof system, a standard hybrid argument shows that the output of $\mathcal{S}$ is computationally indistinguishable from the commit-stage output of R^* in an honest execution. Thus, for every admissible trajectory $\{\text{Sim}(1^\lambda)\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \{\rho_{R^*} : \rho_{R^*} \leftarrow \langle C_{\text{st}}, R^* \rangle(1^\lambda)\}_{\lambda \in \mathbb{N}}$, and the resulting protocol Prot. 1 is computationally hiding.

Computational binding. We prove the unentangled and entangled cases using the same reduction, always taking the adversarial attacking Prot. 1 to be from the same class against Π_{PV} satisfies Def. 10.

Let C^* be a coalition of malicious QPT committers, and suppose that it violates computational binding with probability p over $(\tau, \rho_R, b_{\text{com}}) \leftarrow \langle C^*[\mathcal{S}^*], R \rangle(1^\lambda)$. By the extractability of the underlying commitment scheme, a QPT extractor obtains the committed value $(\text{st}^*, \{\text{sk}_{v,u}\}_{v \in [n], u \in [|\mathcal{L}|]})$, except with negligible probability. Moreover, the binding property of ExtCom ensures that C^* cannot successfully open the commitment to any trajectory $\text{st} \neq \text{st}^*$, except with negligible probability. Hence, Condition 3b in Def. 12 holds, and a violation of computational binding implies that $b_{\text{com}} = 1$ and $\text{st}^* \not\subseteq \mathcal{S}^*$ with probability at least $p - \text{negl}(\lambda)$.

We construct a QPT adversary $\mathcal{A}$ against the soundness of the underlying spatiotemporal position verification protocol. The adversary $\mathcal{A}$ extracts $(\text{st}^*, \{\text{sk}_{v,u}\}_{v \in [n], u \in [|\mathcal{L}|]})$ by the extractor of the underlying extractable post-quantum commitment scheme. Since $\text{st}^* \not\subseteq \mathcal{S}^*$, there exists a space-time point $(L_u, T) \in \text{st}^* \setminus \mathcal{S}^*$ can pass the verification. The adversary forwards the corresponding challenges generated by the honest verifiers for $\Pi_{\text{PV}}^{t_v}$ to C^* , receives the corresponding ciphertext $c_{u,v}$, decrypts it and forwards the recovered response to the verifiers while preserving the required timing.

If $b_{\text{com}} = 1$, then, except with the negligible soundness error of the zero-knowledge proof system, the response is valid. Thus, with probability at least $p - \text{negl}(\lambda)$, $\mathcal{A}$ convinces the honest verifiers that it occupies the spatiotemporal point (L_u, T) , even though $(L_u, T) \notin S^*$, meaning that C^* does not occupy this point. By the security of the underlying position-verification protocol, this probability must be negligible. It follows that $p = \text{negl}(\lambda)$, and hence the spatiotemporal trajectory commitment scheme is computationally binding.

Extractability. Similar to the above, we prove the unentangled and entangled cases using the same reduction. For the extractability property, we prove both *simulation* and *spatiotemporal extraction* properties.

Simulation. Let $C^*[S^*]$ be any malicious committer whose coalition occupies the set of spatiotemporal points S^* during the commit-stage execution. We construct a two-stage extractor $\text{Ext} = (\text{Ext}_0, \text{Ext}_1)$ satisfying the simulation requirement. The first stage $\text{Ext}_0(1^\lambda)$ invokes the extractor of the underlying extractable post-quantum commitment scheme against $C^*[S^*]$. Let $(\{\text{sk}_{v,u}\}_{v \in [n], u \in [L]}, \text{st}^*)$ denote the value extracted from the commitment. If the commitment has no unique valid opening of this form, Ext_0 sets $\text{st}^* := \perp$. It then outputs (st^*, ρ) , where the residual state ρ is all information required to continue the interaction with C^* . This extraction is performed without placing any component of Ext_0 at a spatiotemporal point outside S^* .

On input $(\text{sim}, 1^\lambda, \text{st}^*, \rho)$, the second stage Ext_1 resumes the interaction with C^* from the residual state ρ and emulates the honest receiver R . In particular, Ext_1 generates and transmits the challenges of the position-verification instances and receives the corresponding encrypted responses. Whenever $\text{st}^* \neq \perp$, it decrypts these responses using the extracted key $(\{\text{sk}_{v,u}\}_{v \in [n], u \in [L]}, \text{st}^*)$ and checks them against the position-verification predicates associated with the extracted trajectory st^* . The extractor performs all remaining verification steps, including the zero-knowledge verification procedure, according to the honest-receiver algorithm, and sets $b_{\text{com}} := 0$ if any of these checks fails. It then outputs $(\tilde{S}_{C^*}, \tilde{b}_{\text{com}}) \leftarrow \text{Ext}_1(\text{sim}, 1^\lambda, \text{st}^*, \rho)$.

We now relate the effective extracted value $\Gamma_{\tilde{b}_{\text{com}}}^-(\text{st}^*)$ to the committed value $\text{val}_\Pi(\tau, \rho_R)$ from [Def. 13](#). If $\tilde{b}_{\text{com}} = 1$, then all verification steps accept, and the soundness of the verification procedures, together with the binding property of the underlying commitment scheme, guarantees that st^* is the unique valid trajectory to which the commitment can be opened, except with negligible probability. Thus, $\text{val}_\Pi(\tau, \rho_R) = \text{st}^* = \Gamma_{\tilde{b}_{\text{com}}}^-(\text{st}^*)$. If $\tilde{b}_{\text{com}} = 0$, then either no unique valid value was extracted or one of the conditions required for a valid opening fails; hence, the real execution has no unique valid committed trajectory and $\text{val}_\Pi(\tau, \rho_R) = \perp = \Gamma_{\tilde{b}_{\text{com}}}^-(\text{st}^*)$. Consequently, except with negligible probability, $\Gamma_{\tilde{b}_{\text{com}}}^-(\text{st}^*) \approx \text{val}_\Pi(\tau, \rho_R)$.

By the simulation guarantee of the underlying extractable post-quantum commitment scheme, the joint distribution of the extracted value and the simulated residual state is computationally indistinguishable from that of the committed value and the corresponding residual state in a real commit-stage execution. Furthermore, Ext_1 continues the interaction by applying the same efficient honest receiver algorithm to the simulated residual state. Hence, by closure of computational indistinguishability under efficient post-processing and the preceding correspondence between the effective extracted value and the committed value, we have the indistinguishability between the simulated output and the real output. Therefore, Ext satisfies the simulation requirement.

Spatiotemporal Extraction. We follow the same reduction as in the simulation argument. The first stage $\text{Ext}_0(1^\lambda)$ invokes the extractor of the underlying extractable post-quantum commitment scheme to obtain the committed value $(\{\text{sk}_{v,u}\}_{v \in [n], u \in [L]}, \text{st}^*)$, together with the auxiliary state ρ required to continue the execution. In the proving mode, $\text{Ext}_1(\text{prv}, 1^\lambda, \text{st}^*, \rho)$ acts as the prover in the

trajectory-verification protocol for st^* . Whenever the verifier sends a challenge corresponding to a point $\alpha \in \text{st}^*$, Ext_1 forwards the challenge to C^* , receives the resulting encrypted response, decrypts it using $\{\text{sk}_{v,u}\}_{v \in [n], u \in [L]}$, and forwards the recovered response to the external verifier. The (simulatable) extractability of ExtCom guarantees that $(\{\text{sk}_{v,u}\}_{v \in [n], u \in [L]}, \text{st}^*)$ is the uniquely committed value and that the post-extraction state of C^* is preserved in a computationally-indistinguishable sense, while the correctness of the XOR operation guarantees that the response recovered by Ext_1 is precisely the response generated by C^* . Moreover, by the soundness of the zero-knowledge proof system, if the proof at the end of the commit phase is accepted, then, except with negligible probability, the corresponding **NP** statement is true. In particular, all conditions encoded in the **NP** relation hold: $\text{st}^* \in \mathbb{T}$, the commitment correctly opens to $(\{\text{sk}_{v,u}\}_{v \in [n], u \in [L]}, \text{st}^*)$, and the decrypted responses is valid for all $\alpha \in \text{st}^*$. Therefore, acceptance of the final proof certifies the validity of the intermediate protocol executions encoded in the relation. Consequently, if C^* is accepted in the commit phase with probability $\delta(\lambda) \geq \kappa(\lambda)$, then Ext_1 successfully convinces the external trajectory verifier, i.e., $b' = 1$, and extracts a trajectory $\text{st}^* \in \mathbb{T}$ with probability $\text{poly}(\delta(\lambda))$, as required.

This finishes the proof of [Lem. 1](#). □

Based on the adversarial model under consideration, we provide two instantiations.

Instantiation against Unentangled Adversaries. The position-verification protocol proposed in [\[LLQ22\]](#) satisfies [Def. 10](#) and is secure against unentangled adversaries. Moreover, assuming QLWE, both the post-quantum extractable commitment and the zero-knowledge argument utilized in [Prot. 1](#) can be instantiated in the CRS model. Therefore, we have the following corollary.

Corollary 2. *In the CRS model, assuming QLWE, there exists an extractable spatiotemporal trajectory commitment scheme that is secure against unentangled QPT adversaries.*

Instantiation against Entangled Adversaries (as Defined in [Sec. 4.1](#)). The position-verification protocol presented in [\[Unr14\]](#) achieves security against entangled adversaries (with polynomial queries) in the quantum random-oracle model (QROM), without any extra assumptions. This protocol satisfies [Def. 10](#). Then, using known QROM based post-quantum extractable commitment and zero-knowledge proof systems, we have the following [Corollary 3](#).

However, we emphasize that this corollary does not follow by simply replacing the commitment and ZK protocols in [Prot. 1](#) with their QROM instantiations. Such a replacement is insufficient because [Step 4](#) of [Prot. 1](#) requires a ZK proof of the validity of the extractable commitment execution in [Step 1](#). In the QROM instantiation, the commitment algorithm accesses a random oracle, so the statement to be proved involves oracle evaluations that cannot be represented by an explicit circuit for the oracle. This issue can be resolved by a straightforward adaptation of the black-box commit-and-prove technique of [\[CCLY22a\]](#). Specifically, one can instantiate [Steps 1](#) and [4](#) jointly using an extractable commit-and-prove protocol adapted from [\[CCLY22a\]](#), which uses the QROM-based extractable commitment only as a black box. This yields the result claimed in [Corollary 3](#).

Corollary 3 (of [Lem. 1](#) and [\[CCLY22a\]](#)). *In the QROM, there exists an extractable spatiotemporal trajectory commitment scheme secure against entangled adversaries (with polynomial queries).*

5 UC-secure Spatiotemporal Multi-party Computation

5.1 The SUC Framework

We present the real-world execution, the ideal process, and the definition of *Spatiotemporal UC* (SUC) realization for spacetime-aware multi-party functionalities. We then state the hybrid-model composition theorem for this notion. The exposition closely follows its “ordinary” UC analogue [Can01, CLOS02, Unr10].

Protocol syntax. Following the standard formulation of protocols as systems of quantum polynomial-time machines, we model a protocol as a collection of such machines, each specifying the program executed by a single party. These machines may exchange quantum information. Adversarial entities are likewise modeled as quantum polynomial-time (QPT) machines, and may receive arbitrary non-uniform auxiliary information.

The basic framework. The security of protocols that carry out a given task (or solve a given protocol problem) is defined in three steps. First, we formalize the execution of a protocol in the presence of an adversary and within a given computational environment; this is referred to as the *real-world model*. Second, we formalize an ideal process for carrying out the task. In this ideal process, the parties do not communicate directly with one another. Instead, they interact with an *ideal functionality*, which can be viewed as an incorruptible trusted party programmed to capture the desired behavior of the task. Finally, a protocol is said to SUC-realize an ideal functionality if its real-world execution *emulates* the corresponding ideal process. We next describe the real-world execution model, the ideal process, and the notion of protocol emulation.

We focus on a computational model intended to capture realistic communication networks. Communication takes place over an asynchronous public network that does not guarantee message delivery. We assume authenticated communication; thus, the adversary cannot modify messages sent by honest parties or inject messages on their behalf. However, the adversary controls message delivery: it may deliver any previously sent message at most once or prevent its delivery altogether. Because the network is asynchronous, messages need not be delivered in the order in which they were sent. Parties may be corrupted adaptively during the execution, after which their behavior may be controlled arbitrarily by the adversary. Upon corrupting a party, the adversary obtains its entire internal state, including all retained information from prior stages of the execution. Finally, all entities involved in the execution are restricted to quantum polynomial-time (QPT), or *feasible*, computation.

Protocol execution in the real-world model. We sketch the execution of a protocol π , represented by quantum polynomial-time machines $P_1, \dots, P_n$, in the presence of an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$, where all machines are parameterized by a security parameter $\lambda \in \mathbb{N}$. The execution proceeds as a sequence of activations, beginning with the activation of $\mathcal{Z}$. During each activation, the active machine may transfer designated classical or quantum registers to another machine. It then enters a waiting state and activates the recipient machine.

Whenever the adversary is activated, it may either deliver a message to a party or corrupt a party. Only messages previously sent by a party may be delivered, and each such message may be delivered at most once. Upon corrupting a party, the adversary obtains access to its entire internal state, including all retained information from prior stages of the execution, and controls all of its subsequent actions. The corrupted party may thereafter behave arbitrarily. In addition, the environment is notified whenever a party is corrupted. If, during its activation, the adversary

delivers a message to an uncorrupted party, that party is activated after the adversary's activation ends. Otherwise, the environment is activated next.

Whenever a party is activated, either upon receiving an input from the environment or upon receiving a message delivered by the adversary, it executes its prescribed code and may send messages to other parties. The protocol execution terminates when the environment completes an activation without sending a message to any other entity. The output of the execution is defined as the output of the environment, which we assume to be a single bit.

In summary, the order of activations is as follows. The environment $\mathcal{Z}$ is always activated first. During its activation, $\mathcal{Z}$ may activate either the adversary $\mathcal{A}$ or a party P_i . If P_i is activated directly by $\mathcal{Z}$, it receives an input of the form (x_i, st_i) , where x_i is the classical input provided by $\mathcal{Z}$ and st_i represents the trajectory of P_i ; during the execution of the protocol, P_i moves along this trajectory st_i . If $\mathcal{A}$ is activated, it may either return control to $\mathcal{Z}$ or activate a party P_i by delivering a message to it. After the activation of P_i is complete, control is always returned to $\mathcal{Z}$. We emphasize that exactly one entity is active at any point in the execution. Furthermore, during each activation, $\mathcal{Z}$ and $\mathcal{A}$ may each activate at most one other entity.

Let $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}(k, z, \bar{r})$ denote the output of the environment $\mathcal{Z}$ in an execution with the adversary $\mathcal{A}$ and parties $P_1, \dots, P_n$ running protocol π , where k is the security parameter, z is the input to $\mathcal{Z}$, and $\bar{r} = (r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, \dots, r_n)$ specifies the randomness of all entities:¹⁶ $r_{\mathcal{Z}}$ for the environment, $r_{\mathcal{A}}$ for the adversary, and r_i for party P_i . Let $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}(k, z)$ denote the random variable induced by $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}(k, z, \bar{r})$ when $\bar{r}$ is chosen uniformly at random.

The ideal process. The security of a multiparty protocol is defined by comparing its execution in the real-world model with an ideal process for carrying out a single instance of the task under consideration. A central component of the ideal process is an ideal functionality that captures the desired behavior, or specification, of the task. The ideal functionality is modeled as a quantum polynomial-time (QPT) machine that interacts with the environment and the adversary according to the process described below. More specifically, the ideal process consists of an ideal functionality $\mathcal{F}$, an ideal-process adversary (or simulator) $\mathcal{S}$, an environment $\mathcal{Z}$ with input z , and dummy parties $\tilde{P}_1, \dots, \tilde{P}_n$.

In our setting, the input associated with a party P_i is of the form (x_i, st_i) , where x_i denotes the logical input of P_i and $\text{st}_i \in \mathsf{T}$ specifies the trajectory along which P_i evolves where T is the set of all admissible spacetime trajectories. Thus, upon receiving (x_i, st_i) , the party processes the logical input x_i while advancing along the trajectory st_i .

As in a protocol execution in the real-world model, the environment is activated first. During each activation, it receives information from the dummy parties and the simulator and may send information to either a single dummy party or the simulator. In particular, the environment may provide a dummy party $\tilde{P}_i$ with an input of the form (x_i, st_i) , instructing it to process the logical input x_i while advancing along the trajectory st_i . Once the environment completes its activation, the entity to which it provided input is activated next.

The dummy parties are fixed, simple quantum polynomial-time machines. Whenever a dummy party is activated with an input, it forwards the corresponding information to the ideal functionality $\mathcal{F}$. Upon completion of the dummy party's activation, the environment is activated. Dummy parties communicate only with $\mathcal{F}$. In principle, messages exchanged between the dummy parties and $\mathcal{F}$ are private and cannot be read by the simulator $\mathcal{S}$. Each such message, however, consists of two components: a public header and private contents. The header is visible to $\mathcal{S}$, whereas the contents

¹⁶ We remark that a quantum machine does not require a random tape, since it can generate randomness through quantum measurements as needed. Nevertheless, we retain random tapes for generality, allowing the model to accommodate settings in which some parties are classical machines.

are hidden from $\mathcal{S}$. The specification of $\mathcal{F}$ determines which information is included in the header and which is included in the contents. Certain information must necessarily appear in the public header. For example, the identity of a party to which $\mathcal{F}$ sends an output must be public so that the output can be delivered. Beyond such necessary information, the specification of $\mathcal{F}$ determines the precise division between the header and the contents.

For all functionalities considered in this work, headers follow a fixed format and specify the type of action, the session identifier, and the identities of the participating parties. For example, a commitment message may have the form $(\text{commit}, \text{sid}, P_i, P_j, b)$, where commit specifies that a commitment is being made, sid is the session identifier, P_i is the committing party, P_j is the receiving party, and b is the committed value. In this example, the public header is $(\text{commit}, \text{sid}, P_i, P_j)$, while the private contents consist solely of b .

Whenever the ideal functionality $\mathcal{F}$ is activated, it processes the information it has received and may send messages to the dummy parties and the simulator. Once the activation of $\mathcal{F}$ is complete, the environment $\mathcal{Z}$ is activated next.

When the adversary $\mathcal{S}$ is activated, it gets information from $\mathcal{F}$. In contrast, $\mathcal{S}$ cannot get the private contents of these messages (unless the recipient of the message of $\mathcal{F}$ or a corrupted party). Likewise, $\mathcal{S}$ can get the public headers of the message intended for $\mathcal{F}$. Then, $\mathcal{S}$ can execute one of the following actions. It may either send information to $\mathcal{F}$'s, deliver a message between $\mathcal{F}$ and P_i 's, or corrupt a party. Upon corrupting a party, both $\mathcal{Z}$ and $\mathcal{F}$ learn the identity of the corrupted party. In addition, the adversary learns all the information of the corrupted party. Finally, the adversary controls the party's action from the time that corruption takes place.

If the adversary delivered a message to an uncorrupted dummy party $\tilde{P}_i$ or to the functionality $\mathcal{F}$ in an activation, then this entity is activated next. Otherwise, the environment $\mathcal{Z}$ is activated next.

As in the real-life model, the protocol execution ends when the environment do not send information to any entity. The output of the protocol execution is one-bit output of $\mathcal{Z}$.

In summary, the order of activations in the ideal model is as follows. As in the real model, the environment $\mathcal{Z}$ is always activated first, and then activates either the adversary $\mathcal{A}$ or some dummy party $\tilde{P}_i$. If the adversary $\mathcal{A}$ is activated, then it either activates a dummy party $\tilde{P}_i$ or the ideal functionality $\mathcal{F}$ by delivering the entity a message, or it returns control to the environment. After the activation of a dummy party or the functionality, the environment is always activated next.

Let $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}(k, z)$ denote the output of the environment $\mathcal{Z}$ after interacting in the ideal process with adversary $\mathcal{S}$ and ideal functionality $\mathcal{F}$, on security parameter k , input z and random input $\bar{r} = r_Z, r_S, r_F$, as described above, where z and r_Z are for $\mathcal{Z}$, r_S is for $\mathcal{S}$, and r_F is for $\mathcal{F}$. Let $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}(k, z, \bar{r})$ denote the random variable describing $\text{IDEAL}_{\mathcal{F}, \mathcal{S}, \mathcal{Z}}(k, z, \bar{r})$ when $\bar{r}$ is uniformly chosen.

SUC security in the ideal process. Having defined the real-life model and the ideal process augmented with spacetime trajectories, we now define what it means for a protocol π to SUC-realize an ideal functionality $\mathcal{F}$. Our definition combines computational indistinguishability between the real and ideal executions with a spatiotemporal verification requirement.

Let $\mathcal{I}$ denote the set of identities of the corrupted parties. The ideal-process simulator consists of two components, $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$. During the execution, $\mathcal{S}_0$ is first invoked and outputs strings $\{(x_i^*, \text{st}_i^*)\}_{i \in \mathcal{I}}$ (they are supposed to be the extracted effective input for each corrupted party) and a residual state ρ (meant to be used by $\mathcal{S}_1$). We denote this procedure by

$$(\{(x_i^*, \text{st}_i^*)\}_{i \in \mathcal{I}}, \rho) \leftarrow \mathcal{S}_0(1^\lambda).$$

The component $\mathcal{S}_1$ operates in two modes: **sim** mode and **prv** mode. The **sim** mode is activated immediately after $\mathcal{S}_0$ produces its output. In this mode, $\mathcal{S}_1(1^\lambda, \{(x_i^*, \text{st}_i^*)\}_{i \in \mathcal{I}}, \rho)$ continues the ideal-world execution. It behaves analogously to an ideal-world simulator in the standard UC framework: it submits a value on behalf of each corrupted party, receives the corresponding output from $\mathcal{F}$, and forwards that output to the environment. In addition, it must obey the following rule:

- For each $i \in \mathcal{I}$, it decides whether to submit (x_i^*, st_i^*) or $\perp$ on behalf of party i through the prescribed ideal-process interface. We emphasize that $\mathcal{S}_1$ cannot modify the extracted values (x_i^*, st_i^*) ; it may only decide whether to forward each pair unchanged to $\mathcal{F}$ or to submit $\perp$ instead.

For every $i \in \mathcal{I}$ and $\text{st} \in \mathbb{T}$, we define

$$\delta(\lambda, i, \text{st}) := \Pr[\mathcal{S}_1 \text{ in sim mode sends } (*, \text{st}) \text{ to } \mathcal{F}, \text{ in name of party } i], \quad (5.1)$$

where $*$ is a wildcard and the probability is taken over all (classical and quantum) randomness in the entire ideal-world execution.

In the **prv** mode, $\mathcal{S}_1$ is supposed to act as the prover in an associated spatiotemporal trajectory verification protocol Π' for the extracted trajectory claims (see Def. 15).

With this setup, we present the formal definition of SUC-realization in Def. 15. In essence, SUC security requires that the real and ideal executions be computationally indistinguishable to every non-uniform QPT environment. In addition, it requires that non-negligible success in the simulation experiment imply non-negligible acceptance in the associated spatiotemporal trajectory verification.

Definition 15 (SUC-realization). *Let π be an n -party protocol executed by n parties with the common input 1^λ . Each party P_i has an input (x_i, st_i) , where $\text{st}_i \in \mathbb{T}$ describes its spacetime trajectory. Let $\mathcal{F}$ be an ideal functionality with the corresponding input and output interfaces. We say that π SUC-realizes $\mathcal{F}$ if, for every non-uniform quantum polynomial-time real-world adversary $\mathcal{A}$ and every non-uniform quantum polynomial-time environment $\mathcal{Z}$, there exists a non-uniform quantum polynomial-time ideal-process simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$, a spatiotemporal verification protocol $\Pi' = (\mathcal{P}', \mathcal{V}', \mathbb{T}')$ (as per Def. 11), and a polynomial $\text{poly}(\cdot)$ satisfying the aforementioned syntax such that the following conditions hold:*

- Let $\mathbf{S}^* = \{(L_i, T_i)\}_i$ denote the set of spatiotemporal points that (some part of) $\mathcal{A}$ has occupied (see the discussion in Rmk. 2) during the execution of the protocol.
- No parts of $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ can occupy any spatiotemporal point outside the set $\mathbf{S}^*$;

1. **Indistinguishability.** *It holds that*

$$\left\{ \text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}}(1^\lambda) \right\}_{\lambda \in \mathbb{N}} \stackrel{c}{\approx} \left\{ \text{IDEAL}_{\mathcal{F}, \mathcal{S}_1(\text{sim}, \{(x_i^*, \text{st}_i^*)\}_{i \in \mathcal{I}}, \rho), \mathcal{Z}}(1^\lambda) : (\{(x_i^*, \text{st}_i^*)\}_{i \in \mathcal{I}}, \rho) \leftarrow \mathcal{S}_0(1^\lambda) \right\}_{\lambda \in \mathbb{N}}.$$

2. **Spatiotemporal verification.** *Let $\kappa(\lambda)$ be a knowledge error. For all $i \in \mathcal{I}$ and all $\text{st}_i^* \in \mathbb{T}$, if $\delta(\lambda, i, \text{st}_i^*) \geq \kappa(\lambda)$, then it holds that*

$$\Pr[b' = 1 : b' \leftarrow \langle \mathcal{S}_1(\text{prv}, \rho), \mathcal{V}' \rangle(1^\lambda, \text{st}_i^*)] = \text{poly}(\delta(\lambda, i, \text{st}_i^*)),$$

where $b' \leftarrow \langle \mathcal{S}_1(\text{prv}, 1^\lambda, \text{st}_i^*, \rho), \mathcal{V}' \rangle(1^\lambda, \text{st}_i^*)$ denotes the execution of Π' between $\mathcal{S}_1(\text{prv}, 1^\lambda)$ acting as the prover and the honest verifier $\mathcal{V}'(1^\lambda, \text{st}_i^*)$, who outputs b' after the interaction.

Composition. We state the spatiotemporal analogue of the standard UC composition theorem. This SUC composition theorem will later be used to reduce the security of [Prot. 3](#) in the real model to its security in the corresponding hybrid model.

Theorem 4 (Spatiotemporal Universal Composition Theorem). *Let Π be multiple-party protocol in the $(\mathcal{F}_1, \dots, \mathcal{F}_m)$ -hybrid model, where $\mathcal{F}_1, \dots, \mathcal{F}_m$ are ideal functionalities used as sub-routines by Π . Suppose that Π SUC-realizes a spacetime-aware functionality $\mathcal{F}$ in the $(\mathcal{F}_1, \dots, \mathcal{F}_m)$ -hybrid model.*

For each $\ell \in [m]$, let π_ℓ be a protocol that SUC-realizes $\mathcal{F}_\ell$. Assume that the composed protocol is subroutine respecting and subroutine exposing. Let

$$\Pi[\pi_1, \dots, \pi_m]$$

denote the protocol obtained from Π by replacing each ideal call to $\mathcal{F}_\ell$ with an execution of π_ℓ .

Then $\Pi[\pi_1, \dots, \pi_m]$ SUC-realizes $\mathcal{F}$ in the real model.

Proof (Sketch). This proof is a rather straightforward adaptation of the proof techniques for the standard (quantum) UC composition theorem [[Can01](#), [Unr10](#)]. We thus only provide a sketch of the argument below.

For the indistinguishability condition, the SUC framework preserves the interface that allows the environment to interact with the adversary. Hence, the standard UC composition argument applies, and indistinguishability is preserved under subroutine replacement.

For the spatiotemporal verification condition, replacing a cryptographic subroutine introduces neither a forbidden communication path nor additional pre-shared entanglement into the position-verification reduction. Therefore, the resulting adversarial strategy remains admissible in the underlying physical model, and the spatiotemporal-verification condition is preserved. □

5.2 SUC Spatiotemporal Commit-and-Prove

In this section, we first provide a formal definition of UC spatiotemporal commit-and-prove and then present its construction.

5.2.1 Definition

We consider a functionality with the following properties:

- It consists of two phases: a **Commit** phase and a **Prove** phase.
- During the **Commit** phase, the prover moves along a trajectory $\mathbf{st}$, which it holds as private input, and commits to $\mathbf{st}$.
- For a public predicate $f(\cdot)$, the **Prove** phase consists of a zero-knowledge argument for the statement $f(\mathbf{st}) = 1$.

The ideal functionality. Let $\mathbb{T}$ be the set of admissible spacetime trajectories. The ideal functionality $\mathcal{F}_{\text{CnP}}^{\text{ST}}$ interacts with parties P and V . P who moves along a trajectory $\mathbf{st}$ commits to the spatiotemporal trajectory $\mathbf{st}$ with $\mathbf{st} \in \mathbb{T}$. V zero-knowledge learns only whether the P moves along the trajectory $\mathbf{st}$ and the corresponding committed values satisfy $f(\mathbf{st}) = 1$.

Figure 3: Ideal Functionality $\mathcal{F}_{\text{CnP}}^{\text{ST}}$

The functionality maintains an initially empty table $\mathcal{D}$.

Commit phase. Upon receiving $(\text{commit}, \text{st}, \text{idx})$ from P , the functionality records (st, idx) in $\mathcal{D}$ and sends $(\text{commit-done}, \text{idx})$ to V .

Prove phase. Let f be a predicate provided as common input. Upon receiving $(\text{prove}, \text{idx})$ from P , the functionality sends $(\text{prove-ok}, \text{idx})$ to V if there exists some st such that $(\text{st}, \text{idx}) \in \mathcal{D}$ and $f(\text{st}) = 1$; otherwise, it sends $(\text{prove-failed}, \text{idx})$ to V .

Remark 4 (Commitments to multiple trajectories). The prover may commit to multiple trajectories by invoking the **Commit phase** repeatedly, using a distinct index for each trajectory. To jointly prove statements about any selected collection of committed trajectories, the prover could a single **Prove phase** to prove that it knows valid openings of all the selected commitments and that the corresponding trajectories jointly satisfy the prescribed predicates.

5.2.2 Construction

We construct a protocol realizing the ideal functionality $\mathcal{F}_{\text{CnP}}^{\text{ST}}$ by modifying the construction in [Prot. 1](#). The construction uses the following building blocks:

- A “nice” position-verification protocol Π_{PV} satisfying [Def. 10](#). For each spacetime point α , let W_α denote the verification predicate.
- A UC-secure commitment scheme $\text{UcCom} = \langle P, V \rangle$, where P acts as the committer and V as the receiver, satisfying [Def. 6](#).
- A UC-secure zero-knowledge proof system $\Pi_{\text{UC-ZK}}$ satisfying [Def. 7](#).

Protocol 2: SUC commit-and-prove of spatiotemporal trajectory

Setup. This phase is identical to the **Setup** phase of [Prot. 1](#). The prover P plays the role of the committer C , while the verifier V plays the role of the receiver R .

Private Input: The prover P follows a trajectory $\text{st} = ((L_1, T_1), (L_2, T_2), \dots, (L_m, T_m)) \in \mathcal{T}$, which it holds as its private input.

Commit Phase. This phase proceeds identically to the **Commit Phase** of [Prot. 1](#), except that ExtCom and Π_{ZK} are replaced by UcCom and $\Pi_{\text{UC-ZK}}$, respectively. The prover P plays the role of the committer C , while the verifier V plays the role of the receiver R .

Prove Phase. The prover P and verifier V receive a polynomial-time computable predicate f . They then execute the $\Pi_{\text{UC-ZK}}$ protocol in which P proves that the committed trajectory st satisfies f ; that is, $f(\text{st}) = 1$.

Output: The verifier V accepts if and only if neither phase aborts and the zero-knowledge proof is accepted. Otherwise, V rejects.

Theorem 5 (SUC Realization of [Prot. 2](#)). Assume that Π_{PV} is a “nice” position-verification protocol, as specified in [Def. 10](#), that is secure against unentangled (respectively, entangled) adversaries. Further, the $\Pi_{\text{UC-ZK}}$ satisfying [Def. 7](#), and the UcCom satisfying [Def. 6](#). Then the protocol in [Prot. 2](#) SUC-realizes the ideal functionality [Fig. 3](#) according to [Def. 15](#) against unentangled (respectively, entangled) QPT adversaries.

Proof. We consider the cases of a corrupted prover and a corrupted verifier separately.

Corrupted prover. The security of this follows directly from the extractability argument in the proof of [Lem. 1](#). Combining this argument with the composition theorem in [Thm. 4](#) establishes the desired security claim.

Corrupted verifier. We establish the security of the commit phase and the prove phase separately. For each case, we construct an ideal-world simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ such that, for every non-uniform QPT real-world adversary $\mathcal{A}$ and every non-uniform QPT environment $\mathcal{Z}$, the output of $\mathcal{Z}$ in the real execution is computationally indistinguishable from its output in the ideal execution with $\mathcal{S}$ and $\mathcal{F}_{\text{CnP}}^{\text{ST}}$. The simulator internally runs $\mathcal{A}$ and simulates the interfaces and messages of the honest parties.

– **Commit Phase.**

1. $\mathcal{S}_0$ sets the extracted trajectory to $\text{st}^* = \perp$ and outputs the corresponding residual state ρ needed to continue the execution.
2. Since $\text{st}^* = \perp$, $\mathcal{S}_1$ does not forward information to the $\mathcal{F}$. It then internally runs $\mathcal{A}$ and emulates the honest prover P . Upon receiving $(\text{commit-done}, \text{idx})$ from $\mathcal{F}_{\text{CnP}}^{\text{ST}}$, indicating that the honest prover has committed under index idx , $\mathcal{S}_1$ samples an independent uniformly random key $\{\tilde{\text{sk}}_{v,u}\}_{v \in [n], u \in [\mathbb{L}]} \leftarrow \{0, 1\}^{\ell(\lambda)}$ and selects an arbitrary fixed dummy trajectory $\tilde{\text{st}} \in \mathbb{T}$ of the appropriate format.
3. $\mathcal{S}_1$ invokes the simulator of the underlying UcCom scheme to generate a simulated commitment $\widetilde{\text{com}}$ corresponding to $(\{\tilde{\text{sk}}_{v,u}\}_{v \in [n], u \in [\mathbb{L}]}, \tilde{\text{st}})$ and sends $\widetilde{\text{com}}$ to $\mathcal{A}$ according to the message schedule prescribed by the protocol.
4. $\mathcal{S}_1$ pretends to be any possible location and returns all dummy ciphertexts $\tilde{c}_{u,v} \leftarrow \tilde{\text{sk}}_{u,v} \oplus \perp$, using an appropriate fixed-length encoding of $\perp$. These ciphertexts are delivered with the same lengths, number, ordering, and timing as the ciphertexts in an honest execution. And then give a zero-knowledge proof of doing a valid behavior.
5. At the end of the commit phase, $\mathcal{S}_1$ outputs $\tilde{\rho} = (\tilde{\mathbb{M}}, \widetilde{\text{com}}, \tilde{\pi})$.

Indistinguishability. We show that no QPT environment $\mathcal{Z}$ can distinguish the real commit phase execution from the ideal execution generated by $\mathcal{S}_1$. In the real execution, each position verification response is masked by a fresh uniformly random one-time pad; hence, replacing the encrypted responses, one at a time, with encryptions of $\perp$ preserves their distribution. We then replace the honest commitment with a simulated commitment to an independently sampled dummy pair. By the property of the UC-secure commitment scheme, UC-secure zero-knowledge and [Thm. 4](#), this replacement is computationally indistinguishable to $\mathcal{Z}$. The resulting hybrid is distributed exactly as the ideal execution generated by $\mathcal{S}_1$. Therefore, for every QPT environment, it can distinguish the real commit phase execution from the ideal execution with only negligible probability.

– **Prove Phase.** The simulator proceeds as follows.

1. $\mathcal{S}_1$ invokes the simulator of the UC-secure zero-knowledge proof system to generate a simulated accepting proof for the prove phase statement associated with the transcript produced during the simulated commit phase. In particular, $\mathcal{S}_1$ need not know the honest trajectory, secret key, or decommitment information. It interacts with $\mathcal{A}$ according to the prescribed message schedule and provides it with the resulting simulated proof.

2. Finally, $\mathcal{S}_1$ records the resulting adversarial state and outputs the view of $\mathcal{A}$ together with the prove-phase outcome. By construction, this outcome agrees with that prescribed by $\mathcal{F}_{\text{CnP}}^{\text{ST}}$.

Indistinguishability. Consider a hybrid in which the real zero-knowledge proof generated using the honest witness is replaced by a simulated proof. By the UC zero-knowledge property of the proof system and [Thm. 4](#), this replacement is computationally indistinguishable to the corrupted verifier and any QPT environment. Combining this property with the indistinguishability of the simulated commitment and encrypted responses established for the commit phase, we conclude that the corrupted verifier’s joint view across the commit and prove phases is computationally indistinguishable from its view in the real execution.

Also, note that in the corrupted-verifier case, $\mathcal{S}_1$ do not need to submit any information to the functionality. No spatiotemporal verification argument is required.

This finishes the proof of [Thm. 5](#). □

Following [Sec. 4.3.3](#), we obtain the following instantiations.

Instantiation. The position-verification protocol of [\[LLQ22\]](#) is secure against unentangled adversaries, whereas the protocol of [\[Unr14\]](#) is secure against entangled adversaries in the QROM; moreover, both satisfy [Def. 10](#). The required UC-secure commitment scheme and zero-knowledge proof system can also be instantiated either in the CRS model under the LWE assumption [\[DFL⁺09, ABKK23\]](#) or in the QROM [\[ABKK23\]](#). We therefore obtain the following corollary.

Corollary 4. *Depending on the adversarial model and setup assumptions, the following instantiations exist:*

- *In the CRS model, assuming QLWE, there exists an SUC-secure commit-and-prove scheme against unentangled QPT adversaries.*
- *In QROM, there exists an SUC-secure commit-and-prove scheme against entangled QPT adversaries.*

Remark 5 (Local vs. Global Setup for UC Security). For UC security with setups like CRS or ROM, we distinguish between local and global setup, following the treatment of globally shared setup in [\[CDPW07, CJS14, CDG⁺18\]](#). In the local-setup model considered here, the setup is a subroutine of the protocol instance under consideration; in a global-setup model, it is also accessible to the environment and other protocol instances. The limitation of local setup is that the resulting security guarantee does not automatically extend to arbitrary external protocols sharing the same CRS or random oracle. In particular, a proof using independent local setup instances does not, by itself, justify replacing them with a single globally shared setup.

Nevertheless, this restriction does not reduce our goal to stand-alone security: quantum UC security still supports modular composition and concurrent execution, provided the composition respects the modeled setup boundaries [\[Unr10\]](#). Achieving this guarantee remains nontrivial, since the simulator must maintain an indistinguishable ongoing interaction with a quantum environment. In our setting, it must additionally support certification of the extracted spatiotemporal claims, as described above. Thus, local setup provides a meaningful setting in which to establish composable spatiotemporal security, while leaving unrestricted setup sharing outside our present scope. We leave extending our construction to an explicitly specified global-setup model for future work.

5.3 Construction: Spatiotemporal MPC

In this section, we present a construction for spatiotemporal multiparty computation (MPC). Each party P_i holds a classical input x_i and follows a spatiotemporal trajectory $\mathbf{st}_i$. The parties jointly evaluate the classical functionality

$$F((x_1, \mathbf{st}_1), \dots, (x_n, \mathbf{st}_n)),$$

while ensuring that the trajectory associated with each MPC message is physically verified at the time the message is transmitted.

The protocol consists of the following two phases:

- **Commit phase.** Each party commits to its trajectory.
- **Compute phase.** The parties execute the MPC protocol in conjunction with the UC-secure commit-and-prove protocol for spatiotemporal trajectories. The MPC protocol evaluates the desired functionality, while the commit-and-prove protocol verifies the physical validity of the trajectories associated with the computation.

Because the MPC protocol may consist of multiple communication rounds, spatiotemporal trajectory verification is performed separately in each round. More precisely, whenever a party sends an MPC message, it must prove the physical validity of the corresponding portion of its trajectory at the time of transmission. The protocol must additionally guarantee that all such per-round trajectories are consistent with the trajectories to which the parties committed during the commit phase. To formalize this consistency requirement, we utilize a commitment-consistent MPC functionality.

Commitment-consistent PQ-MPC (CC-MPC). Let $C = (C_1, \dots, C_n)$ be the public commitment vector, where $C_i = \text{Com}(\mathbf{st}_i; r_i)$. The commitment-consistent MPC functionality $\mathcal{F}_{\text{CC-MPC}}$ receives from every party P_i a private input $(x_i, \mathbf{st}_i, \text{decom}_i)$. It checks that

$$\text{Verify}(C_i, \mathbf{st}_i, \text{decom}_i) = 1$$

for every $i \in [n]$. If any check fails, it outputs **abort**. Otherwise, it outputs

$$F((x_1, \mathbf{st}_1), \dots, (x_n, \mathbf{st}_n)).$$

Thus, the functionality guarantees that the trajectories used as inputs to the MPC computation are exactly those fixed by the public commitments.

Round structure and Time Framing. Recall that, in our treatment to position/trajectory verification, we assume that local computations are performed instantaneously. This assumption is reasonable because some position/trajectory-verification protocols require only simple local computations, such as measuring qubits in the computational or Hadamard basis. Moreover, even when local computations are not instantaneous, one can typically adjust the verifiers' decision function to account for their duration, under a reasonable, commonly agreed-upon upper bound on the local computation time.

For MPC protocols, however, we choose to make the local computation time explicit for the following reasons: Depending on the target functionality, the local computations in an MPC protocol may be considerably more involved than those in a position/trajectory-verification protocol, and their duration may therefore be significant. Furthermore, modifying the decision function to account for local computation time may become increasingly complicated as the number of parties grows and

the network topology becomes more complex. We therefore explicitly account for local computation time in our definition of spatiotemporal MPC.

To this end, we assume that each round r of an MPC protocol is executed within a time interval $(t_{r-1}, t_r]$ agreed upon by all parties at the beginning of the protocol. This time interval is further divided into two sub-intervals using a parameter Δ_{proof} , which denote the execution time of the prove stage of the trajectory commit-and-prove protocol. Looking forward in [Prot. 3](#):

- During the time interval $(t_{r-1}, t_r - \Delta_{\text{proof}})$, each party performs its local computation for round r . At time $t_r - \Delta_{\text{proof}}$, all parties send their messages to the designated recipients or broadcast them over the public channel. In particular, even if a party completes its local round- r computation before $t_r - \Delta_{\text{proof}}$, it does not send its messages until time $t_r - \Delta_{\text{proof}}$. This convention ensures that all round- r messages are sent simultaneously, allowing their spatiotemporal verification to be performed at a common time point.
- During the time interval $(t_r - \Delta_{\text{proof}}, t_r]$, each party performs the prove stage of the trajectory commit-and-prove protocol to verify the physical validity of its spatiotemporal trajectory during $(t_{r-1} - \Delta_{\text{proof}}, t_r - \Delta_{\text{proof}}]$.

We will utilize an L -round post-quantum MPC protocol CC-MPC that UC-realize the functionality $\mathcal{F}_{\text{CC-MPC}}$ defined above. The messages sent in round $r \in [L]$ may depend only on the parties' inputs, local randomness, and messages received in rounds $1, \dots, r-1$. We associate the L rounds with a strictly increasing sequence of times $t_0 < t_1 < \dots < t_L$, and define

$$I_{r,\text{comp}} := (t_{r-1}, t_r - \Delta_{\text{proof}}], \quad I_{r,\text{proof}} := (t_r - \Delta_{\text{proof}}, t_r] \quad (5.2)$$

as the time intervals corresponding to round r .

We assume that the interval $[t_0, t_L]$ is a superset of the entire time span of all parties' spatiotemporal trajectories. This is necessary because, as explained in [Sec. 2.2](#), a spatiotemporal computation providing physical authentication guarantees cannot compute a future time point that lies beyond the duration of the protocol execution. This assumption can be achieved even if the original protocol CC-MPC can be executed over a shorter time interval: we can deliberately introduce a delay into each interval I_r , thereby extending $[t_0, t_L]$ to cover the entire time span of all parties' spatiotemporal trajectories. Specifically, the delay can be implemented by requiring each party to wait for a predetermined amount of time before sending its messages in each round.

For each party P_i that sends a message in round r , let

$$\text{st}_{i,r,\text{comp}} := \text{st}_i|_{I_{r,\text{comp}}}, \quad \text{st}_{i,r,\text{proof}} := \text{st}_i|_{I_{r,\text{proof}}} \quad (5.3)$$

denote the restriction of its trajectory to $I_{r,\text{comp}}$ and $I_{r,\text{proof}}$, respectively. The consistency predicate for round r ensures that the trajectory segment used in the corresponding commit-and-prove instance is precisely the restriction of the trajectory committed by P_i during the initial commit phase. Consequently, all per-round physical verifications are tied to a single globally committed trajectory.

Building blocks. The construction uses the following building blocks:

- a post-quantum commitment scheme Com (e.g., Naor's commitment).
- a commit-and-prove of spatiotemporal trajectory protocol TCnP that SUC-realizes the functionality [Fig. 3](#). Let Δ_{proof} denote the maximum time required to complete the prove stage of TCnP.

- a post-quantum MPC protocol CC-MPC in L rounds that UC-realizes the classical functionality $\mathcal{F}_{\text{CC-MPC}}$.

Protocol 3: SUC Spatiotemporal MPC

Setup. The protocol is executed by n parties and is parameterized by a security parameter 1^λ and a nonempty set $\mathbf{T}$ of admissible spatiotemporal trajectories. The execution of CC-MPC spans the entire time interval $[t_{\min}(\mathbf{T}), t_{\max}(\mathbf{T})]$.

Each party P_j is assisted by two coordinating servers that act as verifiers and perform the required spatiotemporal verification and computation. One of these servers is designated as the main server; it verifies the spatiotemporal proofs submitted by the other parties and exchanges CC-MPC messages with their main servers. Meanwhile, P_j moves along a specific trajectory and acts as a prover, proving the validity of its spatiotemporal trajectory to the other parties without revealing the trajectory itself. Consequently, the exchange of CC-MPC messages reveals no spatiotemporal information about P_j .

Private input: The protocol runs by n parties $\{P_j\}_{j \in n}$. Each party P_j holds an input $x_j \in \{0, 1\}^*$ and moves along with a claimed spatiotemporal trajectory $\mathbf{st}_j \in \mathbf{T}$.

Phase 1: Commitment. The parties interact so that each party P_j commits to its trajectory $\mathbf{st}_j$ using the post-quantum commitment scheme Com . We denote the commitment generated by P_j as $C_j = \text{Com}(\mathbf{st}_j; r_j)$. Let $C = (C_1, \dots, C_n)$ denote the public commitment vector. Each party P_j stores its own trajectory $\mathbf{st}_j$ and the corresponding decommitment information decom_j for later use in the MPC execution.

Phase 2: Concurrent MPC. The underlying protocol CC-MPC proceeds according to its prescribed rounds $r = 1, \dots, L$. At each round r , the protocol proceeds according to the following interval schedule:

- **(PQ-MPC execution for round r .)** During the time interval $(t_{r-1}, t_r - \Delta_{\text{proof}})$: each party P_j 's main server computes P_j 's round- r message $m_r^{(j)}$ based on its input x_j , local randomness, and the messages received in all previous rounds. The messages are sent at time point $t_r - \Delta_{\text{proof}}$ to the designated main servers.
In parallel, during the round- r computation time interval $I_{r,\text{comp}}$ as defined in Eq. (5.2), each party P_j acts the committer to execute the commit stage of a TCnP instance, committing to the trajectory segment $\mathbf{st}_{j,r,\text{comp}}$ defined in Eq. (5.3).
- **(TCnP prove phase for round r .)** During the time interval $(t_r - \Delta_{\text{proof}}, t_r]$: each party P_j acts as the prover to execute the prove stage of the TCnP instance for round r , proving that its trajectory segments $\mathbf{st}_{j,r-1,\text{proof}}$ and $\mathbf{st}_{j,r,\text{comp}}$, committed using the commit phase of TCnP during $(t_{r-1} - \Delta_{\text{proof}}, t_{r-1}]$ and $(t_{r-1}, t_r - \Delta_{\text{proof}}]$ respectively, are consistent with the trajectory $\mathbf{st}_j$ committed during the initial commit phase. The main server of each party verifies the proof and, if accepted, allows the corresponding round- r message $m_r^{(j)}$ to be used in subsequent steps. If any proof fails, the protocol aborts.
In parallel, during this round- r (for $r = 1, \dots, L - 1$) proof time interval $I_{r,\text{proof}}$ as defined in Eq. (5.2), each party P_j acts the committer to execute the commit stage of a TCnP instance, committing to the trajectory segment $\mathbf{st}_{j,r,\text{proof}}$ defined in Eq. (5.3).

We note that when $r = L$, we do not need to run the commit sage to commit to $\text{st}_{j,L,\text{proof}}$. This is because the last round message $m_L^{(j)}$ is already sent at time $t_L - \Delta_{\text{proof}}$, representing the end of the real computation. No st_j should contain any (L, T) with $T > t_L - \Delta_{\text{proof}}$.

Output. If all required TCnP instances accept and CC-MPC terminates with local output y_j , party P_j outputs y_j . Otherwise it outputs **abort**.

Theorem 6 (SUC-realization of Prot. 3). *Let $\mathcal{F}_{\text{MPC}}^{\text{ST}}$ denote the spatiotemporal MPC functionality that computes $F((x_1, \text{st}_1), \dots, (x_n, \text{st}_n))$. Suppose that TCnP SUC-realizes the ideal functionality in Prot. 2 and that CC-MPC UC-realizes the functionality $\mathcal{F}_{\text{CC-MPC}}$. Then the protocol in Prot. 3 SUC-realizes $\mathcal{F}_{\text{MPC}}^{\text{ST}}$.*

Proof. Indistinguishability. We prove indistinguishability by a sequence of hybrid games. The first hybrid $\mathcal{H}_0$ is the real execution of Prot. 3, while the final hybrid $\mathcal{H}_3$ is the ideal execution with $\mathcal{F}_{\text{MPC}}^{\text{ST}}$. In every hybrid, the environment $\mathcal{Z}$ interacts with the same externally visible interfaces, and an abort in any invoked subprotocol causes the entire execution to abort. In each hybrid, we describe the differences from the previous hybrid.

- **Hybrid $\mathcal{H}_0$.** This is the real execution of Protocol 3.
- **Hybrid $\mathcal{H}_1$.** In this hybrid, the execution of CC-MPC is replaced by an invocation of $\mathcal{F}_{\text{CC-MPC}}$. Phase 1 and every TCnP instance remain real.
- **Hybrid $\mathcal{H}_2$.** In this hybrid, we additionally replace each round- r instance of TCnP with an invocation of $\mathcal{F}_{\text{CnP}}^{\text{ST}}$. Upon receiving a prove request for predicate g_r corresponding to the statement proven by the round- r proof, the functionality accepts only if the committed trajectory segment is physically valid and satisfies g_r . In particular, any accepted segment must be consistent with the trajectory committed by the sender during the commit phase.
- **Hybrid $\mathcal{H}_3$.** This is the ideal execution with $\mathcal{F}_{\text{MPC}}^{\text{ST}}$.

We now prove indistinguishability between each pair of consecutive hybrids.

- $\mathcal{H}_0 \stackrel{c}{\approx} \mathcal{H}_1$: The only difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is that the real execution of CC-MPC is replaced by an invocation of $\mathcal{F}_{\text{CC-MPC}}$ together with its UC simulator. Since CC-MPC UC-realizes $\mathcal{F}_{\text{CC-MPC}}$, and the surrounding protocol satisfies the required subroutine-respecting and subroutine-exposing conditions, Thm. 4 implies that this replacement remains computationally indistinguishable in the complete spatiotemporal execution. Hence, $\mathcal{H}_0 \stackrel{c}{\approx} \mathcal{H}_1$.
- $\mathcal{H}_1 \stackrel{c}{\approx} \mathcal{H}_2$: The two hybrids differ only in that each real instance of TCnP in $\mathcal{H}_1$ is replaced in $\mathcal{H}_2$ by an invocation of $\mathcal{F}_{\text{ST-CnP}}$ together with the corresponding SUC simulator. Consider a sequence of intermediate hybrids that replaces the TCnP instances one at a time. By the SUC security of TCnP and the universal composition guarantee of Thm. 4, every pair of adjacent hybrids is computationally indistinguishable. Since the protocol invokes only polynomially many TCnP instances, a standard hybrid argument yields $\mathcal{H}_1 \stackrel{c}{\approx} \mathcal{H}_2$.
- $\mathcal{H}_2 \stackrel{c}{\approx} \mathcal{H}_3$: We construct an ideal-world simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1)$ such that the ideal execution with $\mathcal{F}_{\text{MPC}}^{\text{ST}}$ is computationally indistinguishable from $\mathcal{H}_2$. Let $\mathcal{A}$ be the adversary in $\mathcal{H}_2$. The simulator $\mathcal{S}$ runs an internal copy of $\mathcal{A}$, simulates the public interfaces of the $\mathcal{H}_2$ execution, and forwards all messages between $\mathcal{Z}$ and the internal copy of $\mathcal{A}$. It also records the public commitment vector $C = (C_1, \dots, C_n)$ generated in Phase 1. At this stage, $\mathcal{S}$ neither learns

nor uses the inputs of the honest parties; the commitment C_i serves only to bind the trajectory used by P_i in the MPC execution to the trajectory segments subsequently verified by the round-specific instances of $\mathcal{F}_{\text{CnP}}^{\text{ST}}$.

For every corrupted party $P_i \in \mathcal{I}$, the simulator proceeds as follows:

1. When the internal adversary supplies $(x_i^*, \text{st}_i^*, \text{decom}_i^*)$ to $\mathcal{F}_{\text{CC-MPC}}$, the simulator emulates the honest parties and checks whether $\text{Verify}(C_i, \text{st}_i^*, \text{decom}_i^*) = 1$. If this check fails, $\mathcal{S}_1$ simulates the corresponding abort. Otherwise, it records (x_i^*, st_i^*) as the input of P_i .
2. Whenever the internal adversary invokes $\mathcal{F}_{\text{CnP}}^{\text{ST}}$ in round r , $\mathcal{S}_0$ extracts and records the trajectory segment submitted by $\mathcal{A}$. The component $\mathcal{S}_1$ then checks the outcome of the ideal functionality and records the extracted trajectory segment if it is accepted. By the definition of the predicate g_r , every accepted segment is the corresponding portion of a trajectory opening the same Phase 1 commitment C_i . Since Com is computationally binding, except with negligible probability, all accepted segments are consistent with the trajectory st_i^* supplied to $\mathcal{F}_{\text{CC-MPC}}$ and with the unique trajectory committed in C_i .

Once the required trajectory segments have been accepted and jointly cover the prescribed execution interval, $\mathcal{S}_1$ submits (x_i^*, st_i^*) to $\mathcal{F}_{\text{MPC}}^{\text{ST}}$ on behalf of $\mathcal{A}$. If any required invocation of $\mathcal{F}_{\text{CnP}}^{\text{ST}}$ rejects, or if the submitted trajectory fails to open C_i , the simulator reproduces the corresponding abort. Otherwise, the corrupted-party inputs submitted to $\mathcal{F}_{\text{MPC}}^{\text{ST}}$ are exactly those accepted by $\mathcal{F}_{\text{CC-MPC}}$ in $\mathcal{H}_2$, while the honest parties supply their inputs directly to the ideal functionality. Consequently, the output returned by $\mathcal{F}_{\text{MPC}}^{\text{ST}}$ is identical to the output returned by $\mathcal{F}_{\text{CC-MPC}}$ in $\mathcal{H}_2$, and $\mathcal{S}_1$ forwards this output to the internal adversary.

It follows that the output and abort behavior in $\mathcal{H}_3$ agree with those in $\mathcal{H}_2$, except with negligible probability arising from a violation of the computational binding security of Com . Therefore, $\mathcal{H}_2 \stackrel{c}{\approx} \mathcal{H}_3$.

Combining the hybrid transitions, we obtain the indistinguishability of the real and ideal executions.

Spacetime verification. Building on the proof that $\mathcal{H}_2 \stackrel{c}{\approx} \mathcal{H}_3$, we establish the [Condition 2](#) of [Thm. 6](#). Consider any tuple $(\text{st}_i^*, x_i^*, \rho)$ extracted by the simulator $\mathcal{S}_0$. Whenever $\mathcal{S}_1$ forwards (st_i^*, x_i^*) to the ideal functionality, $\mathcal{F}_{\text{CnP}}^{\text{ST}}$ must have received the segment of (st_i^*) from its internal simulator. By the definition of $\mathcal{F}_{\text{CnP}}^{\text{ST}}$, we may invoke its simulator $\mathcal{S}_2^i$ on each segment of st_i^* , thereby verifying every corresponding segment. Consequently, if $\mathcal{S}_1$ forwards (st_i^*, x_i^*) to the ideal functionality with probability $\delta(\lambda)$, then the extracted trajectory st_i^* satisfies the required spacetime-verification condition with probability $\delta(\lambda)$. This completes the proof. $\square$

Instantiation. By [Corollary 4](#), an SUC-secure commit-and-prove scheme can be instantiated against unentangled QPT adversaries in the CRS model under the LWE assumption and against entangled QPT adversaries in the QROM. Moreover, the required post-quantum commitment scheme and UC-secure zero-knowledge proof system can be instantiated in the respective models [[DFL⁺09](#), [ABKK23](#)]. We therefore obtain the following corollary (see also [Rmk. 5](#)).

Corollary 5. *Depending on the adversarial model and setup assumptions, the following instantiations exist:*

- In the CRS model, assuming QLWE, there exists an SUC-secure spatiotemporal MPC protocol against unentangled QPT adversaries.
- In QROM, there exists an SUC-secure spatiotemporal MPC protocol against entangled QPT adversaries.

6 Spatiotemporal Multiparty Quantum Computation

In this section, we extend our construction to quantum functionalities. The main difference from the classical setting is that each party’s input is a quantum–classical pair $(\mathbf{x}_i, \mathbf{st}_i)$, where $\mathbf{x}_i$ is a quantum register and $\mathbf{st}_i$ remains classical. Accordingly, the desired functionality $\mathcal{Q}$ may perform quantum computation on the registers $\mathbf{x}_1, \dots, \mathbf{x}_n$ and classical computation on the trajectories $\mathbf{st}_1, \dots, \mathbf{st}_n$. We therefore employ an MPQC protocol capable of evaluating such a hybrid quantum–classical functionality.

Because the trajectory components remain classical throughout the execution, their physical validity and commitment consistency can be enforced using the same approach as in [Prot. 3](#). Specifically, each party P_i commits to its classical trajectory $\mathbf{st}_i$, and the MPQC protocol evaluates a modified functionality $\mathcal{Q}'$. This functionality first verifies that each supplied trajectory, together with its corresponding decommitment, correctly opens the associated commitment. If all commitment checks succeed, it evaluates

$$\mathcal{Q}((\mathbf{x}_1, \mathbf{st}_1), \dots, (\mathbf{x}_n, \mathbf{st}_n));$$

otherwise, it outputs $\perp$. Concurrently, the TCnP protocol verifies the physical validity of each classical trajectory and proves that the physically verified trajectory is consistent with the trajectory bound to the corresponding commitment. This is performed in exactly the same manner as in [Prot. 3](#).

Thus, the MPQC protocol performs the desired computation, whereas TCnP provides physical verification of the classical trajectory components. Since the trajectories remain classical, the commitment-consistency mechanism from the classical construction applies without modification.

On the SUC framework for MPQC. Technically speaking, the SUC framework we defined in [Def. 15](#) is applicable to classical functionalities. However, the framework can be adapted to quantum functionalities with only minor modifications. In particular, the real-world execution, ideal process, and notion of SUC realization are defined as in [Def. 15](#), except that each party’s input register $\mathbf{x}_i$ may now contain a quantum state, while its trajectory $\mathbf{st}_i$ remains classical. Accordingly, the ideal functionality may receive, process, and output quantum states. Since all other definitions and components remain unchanged from the classical framework, we do not restate them here and throughout use the notion of SUC realization given in [Def. 15](#).

7 Acknowledgements and AI Disclosure

Acknowledgements. Fuyuki Kitagawa and Xiao Liang would like to thank Tomoyuki Morimae, the organizer of the *2nd Kyoto Quantum Crypto Workshop* [[YIT25](#)]. The idea for this work was first conceived and discussed during their participation in the workshop.

AI Disclosure. Large Language Models were used solely to polish the language and improve the clarity of certain parts of the manuscript. They were not used to generate paragraphs or to develop any of the ideas, techniques, or proofs presented in this work.

References

- ABG⁺21. Amit Agarwal, James Bartusek, Vipul Goyal, Dakshita Khurana, and Giulio Malavolta. Post-quantum multi-party computation. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 435–464, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland. [7](#)
- ABG26. Prabhanjan Ananth, Divyanshu Bhardwaj, and Aparna Gupte. Non-interactive MPC, revisited. Cryptology ePrint Archive, Paper 2026/302, 2026. The 24th Theory of Cryptography Conference (TCC 2026). [7](#)
- ABKK23. Amit Agarwal, James Bartusek, Dakshita Khurana, and Nishant Kumar. A new framework for quantum oblivious transfer. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 363–394, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. [19](#), [20](#), [40](#), [45](#)
- Azu97. Ronald T Azuma. A survey of augmented reality. *Presence: teleoperators & virtual environments*, 6(4):355–385, 1997. [2](#)
- Bar21. James Bartusek. Secure quantum computation with classical communication. In Kobbi Nissim and Brent Waters, editors, *TCC 2021: 19th Theory of Cryptography Conference, Part I*, volume 13042 of *Lecture Notes in Computer Science*, pages 1–30, Raleigh, NC, USA, November 8–11, 2021. Springer, Cham, Switzerland. [7](#)
- BCF⁺11. Harry Buhrman, Nishanth Chandran, Serge Fehr, Ran Gelles, Vipul Goyal, Rafail Ostrovsky, and Christian Schaffner. Position-based quantum cryptography: Impossibility and constructions. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 429–446, Santa Barbara, CA, USA, August 14–18, 2011. Springer Berlin Heidelberg, Germany. [2](#), [6](#)
- BCF⁺14. Harry Buhrman, Nishanth Chandran, Serge Fehr, Ran Gelles, Vipul Goyal, Rafail Ostrovsky, and Christian Schaffner. Position-based quantum cryptography: Impossibility and constructions. *SIAM Journal on Computing*, 43(1):150–178, 2014. [2](#)
- BCKM21a. James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. On the round complexity of secure quantum computation. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 406–435, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. [16](#), [17](#)
- BCKM21b. James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. One-way functions imply secure computation in a quantum world. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 467–496, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. [6](#), [7](#), [11](#)
- BCM⁺18. Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 320–331, Paris, France, October 7–9, 2018. IEEE Computer Society Press. [1](#)
- BCS22. Andreas Blum, Matthias Christandl, and Florian Speelman. A single-qubit position verification protocol that is secure against multi-qubit attacks. *Nature Physics*, 18(6):623–626, 2022. [7](#), [8](#)
- BHOY26. James Bartusek, Zikuan Huang, Leo Orshansky, and Henry Yuen. How to track qubits through space and time (or: Sailing in a quantum boat). *arXiv preprint arXiv:2605.30732*, 2026. [3](#), [6](#), [8](#), [9](#), [12](#), [25](#)
- BK02. Mark Billinghurst and Hirokazu Kato. Collaborative augmented reality. *Communications of the ACM*, 45(7):64–70, 2002. [2](#)
- BK11. Salman Beigi and Robert König. Simplified instantaneous non-local quantum computation with applications to position-based cryptography. *New Journal of Physics*, 13(9):093036, 2011. [6](#)
- BLW08. Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In Sushil Jajodia and Javier López, editors, *ESORICS 2008: 13th European Symposium on Research in Computer Security*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206, Málaga, Spain, October 6–8, 2008. Springer Berlin Heidelberg, Germany. [7](#)
- BS20. Nir Bitansky and Omri Shmueli. Post-quantum zero knowledge in constant rounds. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd Annual ACM Symposium on Theory of Computing*, pages 269–279, Chicago, IL, USA, June 22–26, 2020. ACM Press. [7](#), [20](#)
- BY22. Zvika Brakerski and Henry Yuen. Quantum garbled circuits. In Stefano Leonardi and Anupam Gupta, editors, *54th Annual ACM Symposium on Theory of Computing*, pages 804–817, Rome, Italy, June 20–24, 2022. ACM Press. [6](#), [16](#)

- Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press. [5](#), [11](#), [20](#), [33](#), [37](#)
- CCLL26. Nai-Hui Chia, Kai-Min Chung, Xiao Liang, and Jiahui Liu. The black-box simulation barrier persists in a fully quantum world. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology – EUROCRYPT 2026, Part VII*, volume 16547 of *Lecture Notes in Computer Science*, pages 416–445, Rome, Italy, May 10–14, 2026. Springer, Cham, Switzerland. [7](#)
- CCLY22a. Nai-Hui Chia, Kai-Min Chung, Xiao Liang, and Takashi Yamakawa. Post-quantum simulatable extraction with minimal assumptions: Black-box and constant-round. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 533–563, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland. [7](#), [11](#), [32](#)
- CCLY22b. Nai-Hui Chia, Kai-Min Chung, Qipeng Liu, and Takashi Yamakawa. On the impossibility of post-quantum black-box zero-knowledge in constant round. In *62nd Annual Symposium on Foundations of Computer Science*, pages 59–67, Denver, CO, USA, February 7–10, 2022. IEEE Computer Society Press. [7](#), [18](#)
- CDG⁺18. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 280–312, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. [40](#)
- CDPW07. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85, Amsterdam, The Netherlands, February 21–24, 2007. Springer Berlin Heidelberg, Germany. [40](#)
- CGMO09. Nishanth Chandran, Vipul Goyal, Ryan Moriarty, and Rafail Ostrovsky. Position based cryptography. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 391–407, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany. [1](#), [2](#), [8](#), [9](#)
- CH22. Nai-Hui Chia and Shih-Han Hung. Classical verification of quantum depth. *arXiv preprint arXiv:2205.04656*, 2022. [1](#)
- CJS14. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014: 21st Conference on Computer and Communications Security*, pages 597–608, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press. [40](#)
- CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. [6](#), [11](#), [15](#), [20](#), [33](#)
- CLPY25. Rohit Chatterjee, Xiao Liang, Omkant Pandey, and Takashi Yamakawa. The round complexity of black-box post-quantum secure computation. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part IV*, volume 16003 of *Lecture Notes in Computer Science*, pages 3–35, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland. [7](#)
- CRGBB20. Esteban Castro-Ruiz, Flaminia Giacomini, Alessio Belenchia, and Časlav Brukner. Quantum clocks and the temporal localisability of events in the presence of gravitating quantum systems. *Nature Communications*, 11:2672, 2020. [6](#)
- CZO⁺23. Edward Chen, Jinhao Zhu, Alex Ozdemir, Riad S. Wahby, Fraser Brown, and Wenting Zheng. Silph: A framework for scalable and accurate generation of hybrid MPC protocols. In *2023 IEEE Symposium on Security and Privacy*, pages 848–863, San Francisco, CA, USA, May 21–25, 2023. IEEE Computer Society Press. [7](#)
- DDN91. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press. [18](#)
- DEK21. Anders P. K. Dalskov, Daniel Escudero, and Marcel Keller. Fantastic four: Honest-majority four-party secure computation with malicious security. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 2183–2200. USENIX Association, August 11–13, 2021. [7](#)
- DFL⁺09. Ivan Damgård, Serge Fehr, Carolin Lunemann, Louis Salvail, and Christian Schaffner. Improving the security of quantum protocols via commit-and-open. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 408–427, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany. [14](#), [19](#), [20](#), [40](#), [45](#)

- DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, Santa Barbara, CA, USA, August 19–23, 2012. Springer Berlin Heidelberg, Germany. 7
- DS21. Siddhartha Das and George Siopsis. Practically secure quantum position verification. *New Journal of Physics*, 23(6):063069, 2021. 6
- EMS⁺17. Paul Erker, Mark T. Mitchison, Ralph Silva, Mischa P. Woods, Nicolas Brunner, and Marcus Huber. Autonomous quantum clocks: Does thermodynamics limit our ability to measure time? *Physical Review X*, 7(3):031022, 2017. 6
- FFN14. Ben Fisch, Daniel Freund, and Moni Naor. Physical zero-knowledge proofs of physical properties. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 313–336, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany. 1
- GCRB19. Flaminia Giacomini, Esteban Castro-Ruiz, and Časlav Brukner. Quantum mechanics and the covariance of physical laws in quantum reference frames. *Nature Communications*, 10:494, 2019. 6
- GGG⁺26. Uma Girish, Greg Gluch, Shafi Goldwasser, Tal Malkin, Leo Orshansky, and Henry Yuen. Private proofs of when and where. In *Annual International Cryptology Conference*, pages 402–432. Springer, 2026. 3, 5, 6, 8, 14, 23, 27, 28
- GKP17. Sanjam Garg, Susumu Kiyoshima, and Omkant Pandey. On the exact round complexity of self-composable two-party computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 194–224, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland. 20
- GLSV21. Alex B. Grilo, Huijia Lin, Fang Song, and Vinod Vaikuntanathan. Oblivious transfer is in MiniQCrypt. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 531–561, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland. 7, 11
- GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. 1, 6, 15
- GS18. Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 468–499, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. 7
- GW98. Eran Gabber and Avishai Wool. How to prove where you are: Tracking the location of customer equipment. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, pages 142–149, 1998. 8
- IKO⁺11. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 406–425, Tallinn, Estonia, May 15–19, 2011. Springer Berlin Heidelberg, Germany. 7
- IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 21–30, San Diego, CA, USA, June 11–13, 2007. ACM Press. 7
- Kel20. Marcel Keller. MP-SPDZ: A versatile framework for multi-party computation. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1575–1590, Virtual Event, USA, November 9–13, 2020. ACM Press. 7
- KMS11. Adrian Kent, William J Munro, and Timothy P Spiller. Quantum tagging: Authenticating location via quantum information and relativistic signaling constraints. *Physical Review A—Atomic, Molecular, and Optical Physics*, 84(1):012326, 2011. 6
- KPB⁺25. Kirsten Kanneworff, Mio Poortvliet, Dirk Bouwmeester, Rene Allerstorfer, Philip Verduyn Lunel, Florian Speelman, Harry Buhrman, Petr Steindl, and Wolfgang Löffler. Towards experimental demonstration of quantum position verification using single photons. *Quantum Science and Technology*, 10(4):045004, 2025. 7
- KZG⁺26. Gautam A Kavuri, Yanbao Zhang, Abigail R Gookin, Soumyadip Patra, Joshua C Bienfang, Honghao Fu, Yusuf Alnawakhtha, Dileep V Reddy, Michael D Mazurek, Carlos Abellán, et al. Quantum position verification with remote untrusted devices. *arXiv preprint arXiv:2601.16892*, 2026. 7
- LLQ22. Jiahui Liu, Qipeng Liu, and Luowen Qian. Beating classical impossibility of position verification. In Mark Braverman, editor, *ITCS 2022: 13th Innovations in Theoretical Computer Science Conference*, volume 215, pages 100:1–100:11, Berkeley, CA, USA, January 31 – February 3, 2022. Leibniz International Proceedings in Informatics (LIPIcs). 6, 13, 21, 32, 40

- LMS22. Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited or: How to do quantum rewinding undetectably. In *63rd Annual Symposium on Foundations of Computer Science*, pages 851–859, Denver, CO, USA, October 31 – November 3, 2022. IEEE Computer Society Press. 7
- LPY23. Xiao Liang, Omkant Pandey, and Takashi Yamakawa. A new approach to post-quantum non-malleability. In *64th Annual Symposium on Foundations of Computer Science*, pages 568–579, Santa Cruz, CA, USA, November 6–9, 2023. IEEE Computer Society Press. 7
- MNPS04. Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay - secure two-party computation system. In Matt Blaze, editor, *USENIX Security 2004: 13th USENIX Security Symposium*, pages 287–302, San Diego, CA, USA, August 9–13, 2004. USENIX Association. 7
- PKY⁺21. Rishabh Poddar, Sukrit Kalra, Avishay Yanai, Ryan Deng, Raluca Ada Popa, and Joseph M. Hellerstein. Senate: A maliciously-secure MPC platform for collaborative analytics. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021: 30th USENIX Security Symposium*, pages 2129–2146. USENIX Association, August 11–13, 2021. 7
- PSSW09. Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 250–267, Tokyo, Japan, December 6–10, 2009. Springer Berlin Heidelberg, Germany. 7
- PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571, Santa Barbara, CA, USA, August 17–21, 2008. Springer Berlin Heidelberg, Germany. 7
- PW83. Don N. Page and William K. Wootters. Evolution without evolution: Dynamics described by stationary observables. *Physical Review D*, 27(12):2885–2892, 1983. 6
- Ros04. Alon Rosen. A note on constant-round zero-knowledge proofs for NP. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 191–202, Cambridge, MA, USA, February 19–21, 2004. Springer Berlin Heidelberg, Germany. 20
- TFKW13. Marco Tomamichel, Serge Fehr, Jędrzej Kaniewski, and Stephanie Wehner. A monogamy-of-entanglement game with applications to device-independent quantum cryptography. *New Journal of Physics*, 15(10):103002, 2013. 6
- Unr10. Dominique Unruh. Universally composable quantum multi-party computation. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 486–505, French Riviera, May 30 – June 3, 2010. Springer Berlin Heidelberg, Germany. 33, 37, 40
- Unr14. Dominique Unruh. Quantum position verification in the random oracle model. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany. 6, 32, 40
- Wat06. John Watrous. Zero-knowledge against quantum attacks. In Jon M. Kleinberg, editor, *38th Annual ACM Symposium on Theory of Computing*, pages 296–305, Seattle, WA, USA, May 21–23, 2006. ACM Press. 20
- WWP22. Jean-Luc Watson, Sameer Wagh, and Raluca Ada Popa. Piranha: A GPU platform for secure computation. In Kevin R. B. Butler and Kurt Thomas, editors, *USENIX Security 2022: 31st USENIX Security Symposium*, pages 827–844, Boston, MA, USA, August 10–12, 2022. USENIX Association. 7
- Yao82. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press. 1
- Yao86. Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 1
- YIT25. YITP Workshop Kyoto Quantum Crypto 2. Yukawa Institute for Theoretical Physics, Kyoto University, October 2025. <https://www2.yukawa.kyoto-u.ac.jp/~tomoyuki.morimae/KyotoQcrypt2/index.php>. 46