A preliminary version of this paper appears in the proceedings of *the 43nd International Cryptology Conference* (Crypto 2023). This is the full version.

On Concurrent Multi-Party Quantum Computation

Vipul Goyal[1], Xiao Liang[2], and Giulio Malavolta[3]

[1] NTT Research & CMU, USA
vipul@cmu.edu
[2] NTT Research, USA
xiao.crypto@gmail.com
[3] Max Planck Institute for Security and Privacy, Germany
giulio.malavolta@hotmail.it

Abstract. Recently, significant progress has been made toward quantumly secure multi-party computation (MPC) in the *stand-alone* setting. In sharp contrast, the picture of *concurrently* secure MPC (or even 2PC), for both classical and quantum functionalities, still remains unclear. Quantum information behaves in a fundamentally different way, making the job of adversaries harder and easier at the same time. Thus, it is unclear if the positive or negative results from the classical setting still apply. This work initiates a systematic study of *concurrent* secure computation in the quantum setting. We obtain a mix of positive and negative results.

We first show that assuming the existence of post-quantum one-way functions (PQ-OWFs), concurrently secure 2PC (and thus MPC) for quantum functionalities is impossible. Next, we focus on the *bounded-concurrent* setting, where we obtain *simulation-sound* zero-knowledge arguments for both **NP** and **QMA**, assuming PQ-OWFs. This is obtained by a new design of simulation-sound gadget, relying on the recent post-quantum non-malleable commitments by Liang, Pandey, and Yamakawa [arXiv:2207.05861], and the quantum rewinding strategy recently developed by Ananth, Chung, and La Placa [CRYPTO'21] for bounded-concurrent post-quantum ZK.

Moreover, we show that our technique is general enough—It also leads to quantum-secure bounded-concurrent coin-flipping protocols, and eventually *general-purpose* 2PC and MPC, for both classical and quantum functionalities. All these constructions can be based on the quantum hardness of Learning with Errors.

Keywords: Concurrent · Secure Computation · Quantum

Table of Contents

1 Introduction

Secure multi-party computation (MPC) [Yao86, GMW87] enables two or more mutually distrustful parties to compute any functionality without compromising the privacy of their inputs. Since its introduction, MPC has soon become a cornerstone of cryptography. Most papers study MPC only in the so-called *stand-alone* setting, which guarantees the privacy of honest parties for a *single* execution of the underlying protocol.

More realistic setting is the *concurrent setting* where parties might participate in multiple session at a time. A broad study of MPC in the concurrent setting was undertaken starting with the work of Feige and Shamir [FS90, Fei90] on *witness-indistinguishable proofs* in the concurrent setting, and Dwork, Naor and Sahai [DNS98] on *concurrent zero-knowledge proofs*. Unfortunately broad impossibility results for concurrent *self composition* [Lin03, Lin04, BPS06] as well as general composition like *universal composability* [Can01, CF01, CKL03] were soon obtained thereafter.

To overcome this limitation, a number of settings were studied to bypass these results, including

– Standard ideal-world security notion but in weaker real-world model: For example, bounded concurrency [Lin03], CRS model [CLOS02], hardware token model [Kat07], etc.

– Weaker notions of security: For example, super-polynomial-time simulation [Pas03, PS04], input indistinguishable computation [MPR06], simulation with the ability to receive multiple outputs [GJO10, GJ13], etc.

Concurrent MPC in the Quantum Era. All the above impossibility results are in the classical setting. However, it is known that quantum information behaves in a fundamentally different way. For example, the no-cloning theorem [WZ82] might allow us to restrict the ability of the adversary to copy messages. *This raises the tantalizing possibility that assuming laws of quantum physics, concurrently secure computation maybe possible after all!* However, one should also note that, e.g., no cloning also makes the design of the simulator harder since the simulator is no longer free to rewind the (quantum) adversary as in the classical setting. *This on the other hand raises the tantalizing possibility that assuming laws of quantum physics, even results in the weaker setting such as bounded-concurrent secure computation maybe impossible to obtain!*

Our goal in this paper is to initiate a systematic study of concurrently secure computation in the quantum setting.

1.1 Our Results

Notation. We call a protocol *post-quantum* if the honest parties and their communication channels are entirely *classical* but the adversary is allowed to be a quantum machine. We use PQ-MPCC (resp. PQ-2PCC) to denote post-quantum multi-party (resp. two-party) secure computation for *classical* functionalities. Similarly, we use MPQC (resp. 2PQC) to denote secure multi-party (resp. two-party) computation for *quantum* functionalities (over quantum channels), where both the honest parties and the adversaries could be quantum machines.

First, we obtain the following no-go theorem ruling out concurrently secure 2PQC protocols. It can be viewed as a generalization of the [BPS06] impossibility to quantum functionalities.

Theorem 1 (Impossibility). *Assuming the existence of PQ-OWFs, it is impossible to build concurrently secure 2PQC (and thus MPQC) protocols (even assuming quantum computation and communication).*

We remark that assuming the existence of post-quantum one-way functions (PQ-OWFs), concurrently secure PQ-2PCC (and thus PQ-MPCC) is impossible. This follows from an observation that the impossibility results in [BPS06] for concurrently secure 2PCC *in the classical world* extend to the post-quantum setting directly. We do not claim this as our contribution. In contrast, the proof of Thm. 1 is not immediate because of quantum computational and communication.

Next, we investigate possibilities in the *bounded-concurrent* model, where a bound m is a priori fixed (so the protocol design can depend on m) and the adversary is allowed to participate in at most m simultaneous executions of the *same* protocol[4]. This model is interesting because it does not rely on any setup assumptions

[4] We follow the convention that the term "bounded-concurrent" actually means bounded *self* composition.

or relaxations of the security (e.g., super-polynomial-time simulation). It has been demonstrated in the classical setting (e.g., [Lin03, PR03, Pas04]) that a crucial step to build composable computation protocols is to obtain *simulation-sound* ZK systems secure in the (bounded) concurrent setting. Simulation soundness [Sah99] is a form of non-malleability; It requires that the soundness of each of the protocols in the (bounded) concurrent setting is preserved *even when the other protocols are simulated at the same time with the roles of the prover and verifier reversed* (see Def. 11). Intuitively, this notion is crucial for composable 2PC/MPC because it provides a tool for the simulator to "cheat" while ensuring that the adversary cannot, in the concurrent setting.

However, bounded-concurrent simulation-sound ZK arguments are not known in the quantum setting. Therefore, we first propose a new approach to build post-quantum bounded-concurrent simulation-sound ZK arguments. At a high-level, we take the bounded-concurrent ZK argument from [ACL21] and build simulation soundness into it. Moreover, we will show that our technique is general enough—It also leads to bounded-concurrent coin-flipping protocols, and eventually general-purpose 2PC and MPC, for both classical and quantum functionalities. (See for Sec. 2 details.) We summarize the positive results in the following theorem.

Theorem 2 (Positive Results). *There exist constructions, secure against quantum-polynomial-time (QPT) adversaries, for the following tasks in the* bounded-concurrent setting*:*

1. *Simulation-sound ZK arguments for* **NP***, based on the* minimal assumption *of PQ-OWFs. Honest parties of this protocol do not need to perform any quantum computation/communication;*
2. *Simulation-sound ZK arguments for* **QMA***, assuming the existence of PQ-OWFs;*
3. *Two-party coin-flipping, multi-party coin-flipping, PQ-2PCC and PQ-MPCC, assuming the quantum hardness of Learning with Errors; Honest parties of these protocols do not need to perform any quantum computation/communication;*
4. *2PQC and MPQC, assuming the quantum hardness of Learning with Errors.*

1.2 Related Work

In the classical setting, Lindell [Lin03] presented the first m-concurrent two-party protocol for any *a priori* fixed m. Pass and Rosen [PR03] then improved Lindell's results from $O(m)$ rounds to constant rounds. Subsequently, Pass [Pas04] presented a constant-round *multi-party* protocol (and under improved assumptions). The state of the art is from [GLPV20], which can be understood as a black-box version[5] of [Pas04].

In the quantum setting, a recent line of research gave a beautiful characterization of *stand-alone* MPC. For classical functionalities, after Watrous' breakthrough work on post-quantum zero-knowledge [Wat06], the works of [DL09, LN11, HSS11] considered variants of quantum-secure computation protocols in the two-party setting. Recently, constant-round PQ-MPC was also achieved [ABG$^+$21, LPY22]. For quantum functionalities, [DNS12] obtained the first 2PQC protocol. Later, MPQC with dishonest majority was obtained [DGJ$^+$20, ACC$^+$21, GLSV21, BCKM21b, BCKM21a, LPY22].

In contrast, the situation of *concurrently secure* MPC in the quantum setting is less satisfactory. The closest work in this regard is the recent results by Ananth, Chung, and La Placa [ACL21], who built a bounded-concurrently secure protocol for the special case of zero-knowledge arguments.[6] In the plain model, important questions regarding the (im)possibility of composable secure two-party/multi-party computation in the quantum world remained open before the current work.

2 Technical Overview

2.1 Overview of [ACL21]

We first recall the bounded-concurrent post-quantum zero-knowledge (PQ-ZK) arguments from [ACL21]. Let $Q(\lambda)$ be a polynomial of the security parameter λ that denotes the number of concurrent sessions. The [ACL21] protocol proceeds in two stages:

[5] [GLPV20] obtained the same results as [Pas04] while making only block-box use of the underlying primitives.

[6] [ACL21] also obtained a zero-knowledge *proof of knowledge* protocol. This protocol is bounded-concurrent ZK, but [ACL21] only established its proof of knowledge property in the *stand-alone* setting.

– **Preamble Stage:** the prover P and verifier V repeat sequentially for $\ell_{\mathsf{slot}} \coloneqq 120Q^7\lambda$ times the following basic slot: P sends a statistically binding commitment SBCom to a random bit a, V replies by sending another random bit b (in plain). Such a slot is said to *match* if the bit a committed in P's SBCom equals to V's bit b.

– **Proof Stage:** P and V run a witness-indistinguishable (WI) argument where P proves to V that *either* the concerned statement x is true (dubbed the true statement) *or* there are more than $\mathsf{Th} \coloneqq 60Q^7\lambda + Q^4\lambda$ slots match from the above stage (dubbed the $\mathsf{trapdoor}$ statement).

The idea behind the Q-concurrent ZK property of this protocol is as follows:[7] A simulator can always rewind a particular slot until it matches. Therefore, if one can find a proper way to rewind the slots so that the $\mathsf{trapdoor}$ statement becomes true for all the sessions, then the simulator can just use the $\mathsf{trapdoor}$ to finish the **Proof Stage**.

However, since the adversary is a quantum machine, finding a proper rewinding strategy in this Q-concurrent setting is not easy. [ACL21] makes use of Watrous' quantum rewinding lemma [Wat06]. Roughly, this lemma allows one to rewind a quantum adversary under the condition that *the decision of rewinding should be (almost) independent of the adversary's internal (quantum) state.* [ACL21] designs a special *block rewinding* strategy as follows: Let T denote the total number of messages across all sessions in the Q-concurrent execution of their protocol. They partition these T messages into $L \coloneqq 24Q^6\lambda$ equal-size blocks $\{B_1, \ldots, B_L\}$. That is, block B_1 contains the first $\frac{T}{L} = 10Q^2$ messages[8], block B_2 contains the next $10Q^2$ messages, and so on (messages are ordered according to their order of appearing in the execution). Note that the adversary can stagger the messages of a particular session across the different blocks such that the first message of a slot is in one block but the second message of this slot could be in a different block.

As the execution goes on, the simulator monitors each block B_j to see if there is a slot fully nested[9] in B_j. If not, it tosses a random coin to decide whether to rewind the execution of the whole block B_j; Otherwise (i.e., there are at least one fully nested slot), it chooses at random a fully nested slot in B_j, and rewind the execution of the whole block B_j iff the chosen slot matches.

Observe that if there is no fully nested slots in a block, it would be rewound with probability exactly $\frac{1}{2}$; Otherwise, it would be rewound with probability $\frac{1}{2} \pm \mathsf{negl}(\lambda)$ (due to the computationally hiding proeprty of SBCom). Thus, any block would be rewound with probability $\frac{1}{2} \pm \mathsf{negl}(\lambda)$, *independent of the adversary's behavior*, thus satisfying the condition of Watrous rewinding lemma, which implies that this block rewinding strategy will not change the adversary's view.

On the other hand, [ACL21] also shows that by their choice of parameters, the above rewinding strategy will make the $\mathsf{trapdoor}$ statement available in each session. Roughly, that is because in each session, there are approximately $\frac{\ell_{\mathsf{slot}}}{2} = 60Q^7\lambda$ matching slots *even if there are no rewindings* (as each slot will *naturally* match with probability $\frac{1}{2} \pm \mathsf{negl}(\lambda)$); Then, by a combinatorial argument, the authors manage to show that the above rewinding strategy will contribute at least $Q^4\lambda$ extra matching slots in each session. Thus, the total number of matching slots in each session will exceed the threshold $\mathsf{Th} = 60Q^7\lambda + Q^4\lambda$. This eventually completes the proof of the Q-concurrent ZK property.

2.2 Getting Simulation Soundness

Our first goal is to build simulation-sound ZK arguments in the bounded-concurrent setting. In this setting, a polynomial $Q(\lambda)$ (of the security parameter λ) is a priori fixed so the protocol design can depend on Q. It considers an adversary $\mathcal{A}$ (dubbed the MIM adversary) participates in $2Q$ instances of the same protocol *simultaneously*; In Q instances (dubbed the *left* sessions), $\mathcal{A}$ controls the verifiers talking to honest provers; In the other Q instances (dubbed the *right* sessions), $\mathcal{A}$ controls the provers talking to honest verifiers. Each of the sessions is associated with an ID (or tag), and no right-session ID is equal to any left-session IDs.[10] Henceforth, we will refer to this setting as the *Q-Q MIM execution* if the Q is known. Simulation soundness requires the existence of a simulator who could simulate the view of $\mathcal{A}$ in this Q-Q MIM execution *without*

[7] The soundness of this protocol is less relevant to this overview.

[8] Assuming the SBCom is non-interactive, each slot consists of two messages. The total number T is then $\ell_{\mathsf{slot}} \cdot 2 \cdot Q = 240Q^8\lambda$, which implies that $\frac{T}{L} = 10Q^2$.

[9] That is, both messages of this slot are contained in block B_j.

[10] Without IDs, a man-in-the-middle attack can not be prevented. See [Pas04] for related discussions.

the witness for any of the left sessions, while ensuring that $\mathcal{A}$ cannot generate a convincing proof for a false statement in any of the right sessions *even in this simulated execution*.

The [ACL21] protocol does not satisfy this requirement, because in the man-in-the-middle setting, when the simulator performs their rewinding strategy to generate matching slots in left sessions, the MIM adversary may also be able to make extra slots match in some right sessions so that she can use the $\mathsf{trapdoor}$ witness to cheat in these right sessions.

Our idea is to equip the slots in [ACL21] with a certain simulation-sound property, while retaining the overall structure of their protocol so that we can re-use their block rewinding strategy. This will eventually yield a protocol so that in the Q-Q MIM setting, a simulator can use the $\mathsf{trapdoor}$ on the left (due to the [ACL21] block rewinding strategy), but the MIM adversary cannot in any of the right sessions (due to the simulation soundness that we will add to the [ACL21] slots.)

Organization. In the sequel, we first give a warm-up discussion in Sec. 2.3 about how to build a certain flavor of simulation soundness into a single [ACL21] slot (i.e., in the 1-1 MIM setting). Next, we show how to generalize this result to the Q-Q MIM setting in Sec. 2.4. Then, we show that this bounded-concurrent simulation-sound component can be used to build more applications including bounded-concurrent simulation-sound ZK arguments for both **NP** and **QMA** (Sec. 2.5), secure 2PC for both classical and quantum functionalities (in Sec. 2.6), and secure MPC for both classical and quantum functionalities (in Sec. 2.7). Finally, we discuss the impossibility of (unbounded) concurrent 2PC for quantum functionalities in Sec. 2.8.

2.3 Simulation-Sound Gadgets: 1-1 MIM Setting

Intuition. In this subsection, we consider the man-in-the-middle execution of the [ACL21] slot. That is, a MIM adversary $\mathcal{A}$ participates in two instances of the [ACL21] slot simultaneously; $\mathcal{A}$ corrupts the sender (the role of the played by the prover in the [ACL21] slot) in one instance (dubbed the *right* slot) and corrupts the receiver (the role of the played by the verifier in the [ACL21] slot) in the other (dubbed the *left* slot). We will modify the [ACL21] slot so that in this MIM execution, the right slot matches with probability almost $\frac{1}{2}$, *even if conditioned on the left* slot *matching*.

To do that, we ask the receiver to commit to the bit b in advance using a post-quantum statistically-binding commitment SBCom, and ask the sender to commit to the bit a using a constant-round post-quantum non-malleable commitment, followed by the receiver's decommitment to b. In the MIM execution, the receiver's SBCom will allow us to learn the bit b *non-uniformly* in the left slot. Then, a non-uniform reduction to the sender's non-malleable commitment will show that the match probability of the right slot cannot deviate from $\frac{1}{2}$ even if conditioned on the left slot matching. This construction is shown in Prot. 1, where ENMC is a post-quantum constant-round statistically-binding commitment that is both *non-malleable* and *extractable*. For a reason that will become clear later, we additionally require ENMC to be *first-message binding*, which roughly means that the first message of ENMC already statistically determines the committed value. The ENMC from [LPY22] satisfies this property (see Rmk. 4).

Protocol 1: A 1-1 Simulation-Sound Gadget

This protocol is between a sender (dubbed S) and a receiver (dubbed R); Both of them take a string $\mathsf{id} \in \{0,1\}^\lambda$ as the common input, denoting the ID associated with this execution.

1. R samples $b \xleftarrow{\$} \{0,1\}$ and commits to it using SBCom.
2. S samples $a \xleftarrow{\$} \{0,1\}$ and commits to it using ENMC, where S and R use id as the ID (or tag) for this ENMC.
3. R sends b together with the decommitment information w.r.t. the SBCom in Step 1.

(Through out this overview, we assume for simplicity that both SBCom and ENMC are *perfectly* binding and that the MIM adversary $\mathcal{A}$ does not abort the execution before it completes. In the main body, we will show that these assumptions can be removed easily.)

Let $\mathcal{T}$ denote the transcript (i.e., all the messages exchanged on both the left and right sides) resulted from the above MIM execution. We claim that

$$\Pr_{\mathcal{T}\leftarrow \mathrm{MIM}}[a = b] = \frac{1}{2}, \tag{1}$$

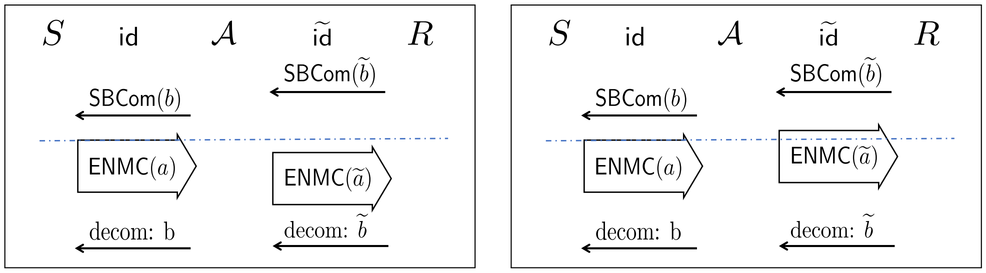


Fig. 1: Different Schedules for the MIM Execution of Prot. 1

where b is the value committed in the left SBCom by $\mathcal{A}$, and a is the value committed in the left ENMC by S. Eq. (1) follows simply from the fact that the left (honest) S samples a uniformly at random, after $\mathcal{A}$'s b committed in SBCom was fixed.

We also claim that

$$\Pr_{\mathcal{T}\leftarrow\text{MIM}}[\widetilde{a}=\widetilde{b}] = \frac{1}{2} \pm \mathsf{negl}(\lambda), \tag{2}$$

where $\widetilde{b}$ is the value committed in the right SBCom by R, and $\widetilde{a}$ is the value committed in the right ENMC by $\mathcal{A}$. Eq. (2) follows from the computationally hiding property of SBCom and the extractability of ENMC. In more details, if Eq. (2) is false, then we can extract the $\widetilde{a}$ (due to the extractability of the right-side ENMC) and break the computationally hiding property of the right-side SBCom using $\widetilde{a}$ as a reasonable guess for the value $\widetilde{b}$ committed in SBCom.

Next, we prove a lemma establishing a certain flavor of simulation soundness of Prot. 1 in the MIM execution. Intuitively, this lemma says that *even if we conditioned on the left side $a = b$*, $\mathcal{A}$ cannot force $\widetilde{a} = \widetilde{b}$ on the right side with probability significantly different from half. This is essentially due to the non-malleability of ENMC.

Lemma 1. *Assuming* $\mathsf{id} \neq \widetilde{\mathsf{id}}$, *it holds that* $\Pr_{\mathcal{T}\leftarrow\text{MIM}}[\widetilde{a}=\widetilde{b} \mid a=b] = \frac{1}{2} \pm \mathsf{negl}(\lambda)$.

Proof. Recall that $\mathcal{A}$ is in charge of the schedule of messages. We use the *first message of the left* ENMC as a pivot to divide all possible schedules into two mutually exclusive and collectively exhaustive types and show Lem. 1 for them using different strategies.

- **Type-1:** They are the schedules where the right ENMC *starts* after (or in parallel with) *the first messages of the left* ENMC. (An example is depicted in Fig. 1a.)
- **Type-2:** They are the schedules where the right ENMC *starts* before *the first message of the left* ENMC. (An example is depicted in Fig. 1b.)

Proof for **Type-1** Schedules. We assume for contradiction that Lem. 1 is false and build an adversary $\mathcal{A}_{\text{NM}}$ who can break the (non-uniform) non-malleability of ENMC. W.l.o.g., we assume

$$\Pr_{\mathcal{T}\leftarrow\text{MIM}}[\widetilde{a}=\widetilde{b} \mid a=b] \geq \frac{1}{2} + \frac{1}{\mathsf{poly}(\lambda)}. \tag{3}$$

A standard calculation with Eq. (1) , Eq. (2) and Inequality (3) implies that

$$\Pr_{\mathcal{T}\leftarrow\text{MIM}}[\widetilde{a}=\widetilde{b} \mid a\neq b] \leq \frac{1}{2} - \frac{1}{\mathsf{poly}(\lambda)} + \mathsf{negl}(\lambda). \tag{4}$$

We now build the adversary $\mathcal{A}_{\text{NM}}$ against the non-malleability of ENMC.

- $\mathcal{A}_{\text{NM}}$ internally emulates the MIM game until the moment when $\mathcal{A}$ sends the left-side SBCom to b. $\mathcal{A}_{\text{NM}}$ then pauses the execution and performs brute-force search to learn the value b. We remark that this step is not efficient. But it happens before the beginning of the ENMCs on both sides (due to **Type-1** schedules). Thus, the information in this step can be thought as a *non-uniform* advice to $\mathcal{A}_{\text{NM}}$.

– $\mathcal{A}_{\rm NM}$ starts to participate in the non-malleability game: She sets $m_0 := b$ and $m_1 := 1 - b$, and sends (m_0, m_1) to the external non-malleability challenger Ch. By the definition of non-malleability, Ch will flip a random coin b and performs a MIM execution of ENMC with $\mathcal{A}_{\rm NM}$ (see Def. 8 for details). $\mathcal{A}_{\rm NM}$ simply relays messages so that these external ENMCs are used as the left and right ENMCs for the internal emulation of the MIM game with $\mathcal{A}$.

Notice that if Ch chooses to commit to m_0 (resp. m_1) on the left, the view of the internal $\mathcal{A}$ is identical to that from the MIM execution conditioned on $a = b$ (resp. $a \neq b$) on the left. Thus, $\mathcal{A}_{\rm NM}$'s advantage in winning the non-malleability game is exactly the difference between the LHS of Inequality (3) and the LHS of Inequality (4), which is *lower-bounded* by $\frac{2}{\mathsf{poly}(\lambda)} - \mathsf{negl}(\lambda)$. This breaks the non-malleability of ENMC.

Proof for **Type-2** Schedules. Security in this case simply follows from the first-message binding property of the right ENMC. That is, the message $\widetilde{a}$ committed in the ENMC is determined by its first message; For **Type-2** schedules, this message is fixed before the beginning of the left ENMC. Therefore, conditioning on $a = b$ on the left does not change the probability of $\widetilde{a} = \widetilde{b}$ for **Type-2** schedules. Lem. 1 then follows directly from Eq. (2) in this case.

This completes the proof of Lem. 1. □

Remark 1. We remark that there is an alternative construction of the 1-1 simulation-sound gadget shown in Prot. 1: One can remove the first $\mathsf{SBCom}(b)$ round, and only ask R to sends the random b after S's ENMC. By a sightly more involved argument, one can prove that this construction satisfies the same 1-1 simulation soundness as Prot. 1. We provide in Appx. A a formal proof of this possibility.

In the sequel, we keep using the current version of Prot. 1 for the following reasons: (1) the version in Appx. A does not have any advantages over Prot. 1 in terms of the required hardness assumptions and the *asymptotic* round complexity; (2) importantly, the security proof for Prot. 1 is conceptually cleaner than that in Appx. A, which requires us to first prove an intermediate lemma regarding the property of non-malleable commitments.

2.4 Simulation-Sound Gadgets: Q-Q MIM Setting

Intuition. In this part, we show that the "simulation soundness" of Prot. 1 extends to the more demanding many-many MIM setting. That is, consider a many-many MIM execution of a protocol consisting of sequential repetitions of the gadget shown in Prot. 1. We want to argue that: even if one performs the [ACL21] block rewinding strategy in this many-many MIM execution to make some gadgets match on the left, it still holds that each gadget (in all the *right* sessions) matches with probability almost $\frac{1}{2}$.

Here, the challenge is that the [ACL21] block rewinding needs to be performed over a coherent execution of the concerned protocol. However, the simulation soundness shown in Lem. 1 is regarding the *de-coherent* execution of a gadget, and in a *straight-line* execution (i.e., where there is no rewindings). Generalizing Lem. 1 to this setting requires new ideas. Roughly, we will introduce extra intermediate hybrids (in addition to those in [ACL21]) to make the execution de-coherent for the blocks that the Watrous rewinding has not reached. Then, by a careful design of induction-type arguments, we manage to reduce the simulation soundness in the Q-Q MIM setting to that in the straight-line 1-1 MIM setting shown in Lem. 1. We provide more details below.

Simulation-Sound Gadgets in the Q-Q MIM Setting. To ensure simulation soundness in the Q-Q MIM setting, we simply repeat the basic gadget in Prot. 1 for sufficiently many times:

Protocol 2: Simulation-Sound Gadgets Secure in the Q-Q MIM Setting

Let $Q(\lambda)$ be a polynomial of λ, denoting the maximum number of concurrent sessions. Both S and R take a string $\mathsf{id} \in \{0,1\}^\lambda$ as the common input.
Repeat the following steps for $k = 1$ to $\ell_{\mathsf{gad}} := 120Q^7\lambda$:

1. R samples $b_k \xleftarrow{\$} \{0,1\}$ and commits to it using SBCom.
2. S samples $a_k \xleftarrow{\$} \{0,1\}$ and commits to it using ENMC, where S and R use $\mathsf{id}{:}k$ as the ID for this ENMC execution so that each ENMC uses a different ID.
3. R sends b_k together with the decommitment information w.r.t. the SBCom in Step 1.

Terminology. We call each repetition of Step 1 to Step 3 a gadget. We say that the k-th gadget *matches* if the a_k committed by S in Step 2 is equal to the b_k *validly* decommitted by R in Step 3. We emphasize that this condition of match is always well-defined regardless of the honesty of S (or R), because both SBCom and ENMC are perfectly binding.

Similar as [ACL21], we partition all the messages into $L \coloneqq 24Q^6\lambda$ equal-size blocks $\{B_1, \ldots, B_L\}$, in the same manner as we explained when recalling the [ACL21] protocol.[11] In the following, we define a sequence of games (or hybrids) G_0 and $\{G_j\}_{j\in[L]}$, the "simulation soundness" of Prot. 2 will appear in the form of two lemmas (Lem. 2 and 3) related to these games.

Game G_0: This is the Q-Q MIM execution of Prot. 2 described above, but executed *coherently*. At the end of this game, we measure (in computational basis) and output the transcript (i.e., all the messages exchanged on both the left and right sides).

Also, we additionally setup L single-qubit registers $\otimes_{j=1}^{L}\mathtt{W}_j$, which will be used for Watrous rewinding only in later games. To explain the meaning of these registers, let us briefly recall the Watrous rewinding lemma. It considers a quantum circuit U operating on the tensor of a single-qubit Watrous control register $\mathtt{W}$ and another multi-qubit register; As long as the output of U induces a distribution on $\mathtt{W}$ that is (almost) independent on the other input register, U can be converted into a new circuit W that on the same input is guaranteed to yield the (almost) same output of U *conditioned on the* Watrous control register *being 0*. Here, each $\mathtt{W}_j$ will be used as the Watrous control register for the rewinding of block B_j. We use a similar strategy as in [ACL21] to ensure that they are (almost) independent of the adversary's behavior: If block B_j does not contain any fully nested gadget from any left sessions (dubbed left gadget), $\mathtt{W}_j$ is set to 0 with probability 1/2; Otherwise (i.e., there exists at least one fully nested left gadget), $\mathtt{W}_j$ is set to 0 iff a randomly-chosen fully nest left gadget in B_j matches.

Game G_j ($\forall j \in [L]$): This game is identical to G_{j-1} except that it uses Watrous rewinding for block B_j, using the $\mathtt{W}_j$ (defined in G_0) as the Watrous control register.

Two Critical Properties of Prot. 2. We prove two lemmas (Lem. 2 and 3) that play a central role in all the positive results in this work.

The following Lem. 2 says that in a hybrid G_j, if a left session is finished before the last messages of block B_j, then the total number of matching gadgets in that session is *large enough*, which means that it exceeds a certain threshold Th that we choose properly.

Lemma 2 (Enough Matching Gadgets). *Let* $\mathsf{Th} \coloneqq 60Q^7\lambda + Q^4\lambda$*. For all* $j \in [L]$*, let* Σ_j *denote the indices of* left *sessions that completes before the last message of block* B_j*. Then, it holds that for all hybrid* G_j $(\forall j \in [L])$ *and all* $i \in \Sigma_j$ *that in hybrid* G_j*, the number of matching* gadgets *in the left session* i *is greater than* Th*, except for with negligible probability.*

(Proof Sketch for Lem. 2). The proof of this lemma is almost identical to [ACL21], where the authors also need to prove that there are enough matching "slots" (i.e., our gadgets in their term). Intuitively, it is because the natural execution will contribute $\frac{\ell_{\mathsf{gad}}}{2} = 60Q^7\lambda$ matching gadgets, and the block rewinding will in additional contribute $Q^4\lambda$ matching ones in each left session. We refer to Appx. B for a formal treatment. □

Definition 1. *For a Q-Q MIM execution of Prot. 2 we say that the* invariant condition *holds iff*

$$\forall i \in [Q], k \in [\ell_{\mathsf{gadget}}], \quad \Pr\left[\widetilde{a}_k^{(i)} = \widetilde{b}_k^{(i)}\right] = \frac{1}{2} \pm \mathsf{negl}(\lambda),$$

where $\widetilde{a}_k^{(i)}$ is the value committed in the k-th ENMC *(i.e., the* ENMC *of the k-th* gadget*) in the i-th right session, $\widetilde{b}_k^{(i)}$ is the value committed in the k-th* SBCom *(i.e., the* SBCom *in the k-th* gadget*) of the i-th right session, and the probability is taken over the Q-Q MIM execution.*

Lemma 3 (Indistinguishability and Invariant Condition). *The view of the adversary in all the games $\{G_0, G_1, \ldots, G_L\}$ are computationally indistinguishable. Moreover, the* invariant condition *(as per Def. 1) holds in all the games, assuming no left-session ID is equal to any right-session ID.*

[11] But notice that we are in the Q-Q MIM setting. Both the left and right sessions contribute messages to blocks.

(Proof Sketch for Lem. 3). We first prove that $\mathcal{A}$'s view is indistinguishable in games $\{G_0, G_1, \ldots, G_L\}$. This follows from a rather straight-forward manner from Watrous' rewinding lemma. To see that, recall that for any $j \in [L]$, the only difference between G_{j-1} and G_j is: G_j will finish block B_j using Watrous rewinding with $\mathtt{W}_j$ playing the role of the Watrous control register. Therefore, the indistinguishability of the views will follow once we show that $\mathtt{W}_i$ is set to 0 (almost) independent of $\mathcal{A}$'s behavior. This simply follows from the way we definition of $\mathtt{W}_j$ (defined in G_0):

1. If there is no fully nested left gadgets in block B_j, $\mathtt{W}_j$ is set to 0 with probability exactly 1/2;
2. Otherwise, the game will pick a fully nested left gadget at random and set $\mathtt{W}_j$ to 0 iff that gadget matches. Note that a left gadget matches with probability exactly 1/2 as well, because the left honest sender $S^{(i)}$ (of the i-th left session) samples the bit $a_k^{(i)}$ uniformly at random (in all left session $i \in [Q]$ and all its gadget $k \in [\ell_{\mathsf{gad}}]$).

Therefore, the Watrous control register is set to 0 with probability exactly 1/2, independent of $\mathcal{A}$'s behavior. This establishes the view indistinguishability of $\mathcal{A}$ in all the games. We note that this proof (for view indistinguishability) is not new to this work. A similar argument already appears in [ACL21].

Invariant Condition. In the following, we prove the invariant condition. *This proof relies on new techniques developed in this work.*

This proof is of the form of mathematical induction. We first establish the invariant condition in game G_0. Next, we show that for all $j \in [L]$, if the invariant condition holds in game G_{j-1}, then it must hold in game G_j as well.

Invariant condition in G_0. In game G_0, there is no Watrous rewinding; All the blocks are executed in straight-line. This makes the proof of invariant condition straightforward—If $\mathcal{A}$ manages to break the invariant condition for the k-th gadget in the i-th right session, then we can extract the $\widetilde{a}_k^{(i)}$ (due to the extractability of ENMC) and break the computationally hiding property of SBCom using $\widetilde{a}_k^{(i)}$ as a reasonable guess for the value $\widetilde{b}_k^{(i)}$ committed in the corresponding SBCom.

One caveat is that we define G_0 to be a *coherent* execution. But the above argument is in the de-coherent setting, where all the messages are classical. This is not a problem because of the following observation: due to the deferred measurement principle, the classical transcript[12] resulted from G_0 is identically distributed as that from the real (de-coherent) Q-Q MIM execution. Since the invariant condition is information-theoretically determined by the transcript, proving it in the de-coherent Q-Q MIM execution is equivalent to proving it in (the coherent) game G_0.

Invariant condition in G_j. Assume that the invariant condition holds in G_{j-1}. We now show that it must hold in G_j as well. Similar as in the above proof for G_0, game G_j is a *coherent* execution. We will instead consider a new game G'_j that *de-coherentizes* the blocks in G_j that are not reached by Watrous rewinding.

- **Game G'_j ($j \in [L]$):** This game is identical to G_j, except that at the end of block B_j, it measures (in computational basis) the transcript so far, and then finish the remaining execution de-coherently.

As we argued before, the invariant condition in G'_j is equivalent to that in G_j, due to the deferred measurement principle. In the following, we focus on an arbitrary gadget $k \in [\ell_{\mathsf{gad}}]$ and right session $i \in [Q]$ in G'_j; we denote this gadget by $\widetilde{\mathsf{gadget}}_k^{(i)}$.

Note that the only difference between G'_{j-1} and G'_j is the Watrous rewinding performed by G'_j for block B_j. By definition, this rewinding could happen for two reasons: (i) there is no left gadget fully nested in block B_j, but we decide to perform a "dummy" rewinding (w.p. 1/2); (ii) there exists at least one left gadget fully nested in block B_j, and the randomly selected gadget does not match. First, note that Case (i) is degenerated, because it corresponds to a "dummy" rewinding that essentially does not change the transcript. For this case, the invariant condition in G'_j is inherited from that in G'_{j-1}. Therefore, in this proof sketch, we only focus on Case (ii).[13]

[12] Recall that the transcript output by G_0 is classical, because G_0 measured it at the end of the execution.
[13] We refer to Sec. 4.5 for a formal treatment of Case (i) and how we combine Cases (i) and (ii).

Recall that the schedule of messages is controlled by $\mathcal{A}$. Similar as in the proof of Lem. 1, we divide the possible schedules into different types and prove the $\mathsf{invariant\ condition}$ for them one by one. This time, we use the last message of block B_{j-1} as the pivot:

Type-1 Schedules: The first message of ENMC to $\widetilde{a}_k^{(i)}$ happens before (or in parallel with) the final message of B_{j-1}. This is an easy case. Note that G'_{j-1} and G'_j are identical until the end of block B_{j-1}, by when the the first message of the $\widetilde{\mathsf{gadet}}_k^{(i)}$ ENMC is already fixed. By the first-message binding property of ENMC, this already fixed the committed value $\widetilde{a}_k^{(i)}$ and thus the $\mathsf{invariant\ condition}$. Therefore, the $\mathsf{invariant\ condition}$ in G'_j is inherited from that in G'_{j-1}.

Type-2 Schedules: The first message of ENMC to $\widetilde{a}_k^{(i)}$ happens after the final message of B_{j-1}. Recall that we are in Case (ii), i.e., G'_j rewinds block B_j because the randomly sampled left gadget (fully nested in B_j) does not match. Let us denote this selected left gadget by $\mathsf{gadget}_v^{(u)}$ (i.e., it happens to be the v-th gadget of some u-th left session).

Here, the key observation is: the transcript resulted from G'_j is actually identical to that from G'_{j-1} *but conditioned on* $\mathsf{gadget}_v^{(u)}$ *matches* (i.e., $a_v^{(u)} = b_v^{(u)}$). This is because that the effect of Watrous rewinding is to "kill" the branch in the superposition that corresponds to the $\mathsf{Watrous\ control\ register}$ being 1, and only retain the branch that corresponds to the $\mathsf{Watrous\ control\ register}$ being 0. It then follows from this observation that proving the $\mathsf{invariant\ condition}$ in G'_j is equivalent to proving that in G'_{j-1} *but conditioned on* $\mathsf{gadget}_v^{(u)}$ *matches*.[14] As a vigilant reader may already notice, this is exactly what we have proven in Lem. 1 for the 1-1 "simulation soundness" of Prot. 1.[15] Thus, the same argument applies here to finish the proof of the $\mathsf{invariant\ condition}$. In more details, we will view all the execution except for $\mathsf{gadget}_v^{(u)}$ and $\widetilde{\mathsf{gadget}}_k^{(i)}$ as a new MIM adversary, and view $\mathsf{gadget}_v^{(u)}$ and $\widetilde{\mathsf{gadget}}_k^{(i)}$ as the left and right $\mathsf{gadgets}$ in the 1-1 MIM setting. This configuration matches exactly the proof of Prot. 1.

This finishes the proof of Lem. 3. □

2.5 Bounded-Concurrent Simulation-Sound ZK Arguments for NP and QMA

We show how to use Prot. 2 to build a bounded-concurrent simulation-sound ZK argument protocol Π_{SSZK} for **NP**. The idea is to first execute a **Preamble** stage where the prover and the verifier run Prot. 2. Then, they will execute a WI argument to prove *either* the concerned statement x is true, *or* there are more than $\mathsf{Th} = 60Q^7\lambda + Q^4\lambda$ matching $\mathsf{gadgets}$ from the execution of Prot. 2 in the **Preamble** stage; An honest prover will use the real witness w (for $x \in \mathcal{L}$) for this stage. This protocol Π_{SSZK} is presented in Prot. 3 below.

Protocol 3: Π_{SSZK}: Q-Q Simulation-Sound ZK Arguments for NP (Informal)

A prover (dubbed P_{id}) and a verifier (dubbed V_{id}) agree on an ID $\mathsf{id} \in \{0,1\}^\lambda$ and an statement x from some **NP** language $\mathcal{L}$; P_{id} additionally holds a witness w for $x \in \mathcal{L}$.

1. **Preamble:** These two parties run Prot. 2 (using id as the ID), where P_{id} acts as the sender S and V_{id} acts as the receiver R.
2. **WI:** Then, they run a WI argument where P_{id} proves that *either* x is in $\mathcal{L}$ *or* there are more than $\mathsf{Th} = 60Q^7\lambda + Q^4\lambda$ matching $\mathsf{gadgets}$ from the execution of Prot. 2 in Stage 1. An honest P_{id} will always use the real witness w (for $x \in \mathcal{L}$) to perform this WI.

To prove simulation soundness of Π_{SSZK} in the Q-Q MIM setting,[16] we need to construct a simulator $\mathcal{S}$ in the Q-Q MIM execution of Π_{SSZK}, who can simulate the view of the MIM adversary $\mathcal{A}$ *without* using the real witness w in any of the Q left sessions; Meanwhile, we need to make sure that even in this simulated execution, $\mathcal{A}$ cannot convince the honest verifier on a false statement in any of the Q right sessions.

Intuitively, the properties (i.e., Lem. 2 and 3) of Prot. 2 performed in the **Preamble** will allow us to construct the desired simulator $\mathcal{S}$ so that $\mathcal{S}$ is able to use the $\mathsf{trapdoor}$ (i.e., more than $\mathsf{Th} = 60Q^7\lambda + Q^4\lambda$

[14] Here, we mean ($\mathsf{invariant\ condition}$ in G'_j) $\equiv$ ($\mathsf{invariant\ condition}$ in G'_{j-1} conditioned on $\mathsf{gadget}_v^{(u)}$ matches).

[15] Note that neither $\mathsf{gadget}_v^{(u)}$ nor $\widetilde{\mathsf{gadget}}_k^{(i)}$ is interleaved with Watrous rewinding in G_{j-1} (due to **Type-2** schedules).

[16] Completeness and soundness of Π_{SSZK} follow from standard techniques. We refer to the main body for details.

gadgets match) to cheat in the **WI** stage against the adversary in each left sessions (due to Lem. 2), and meanwhile ensure that the adversary cannot use the trapdoor and must behave honestly in all right sessions (due to the invariant condition[17] claimed in Lem. 3). However, we cannot use Lem. 2 and 3 *in a modular way*, because we only proved them for the Q-Q MIM execution of Prot. 2 *itself*, where there are no other protocols. It is unclear if they still hold when Prot. 2 is composed with other messages (i.e., the **WI** stage messages). Nevertheless, we will show in the following that the proofs of Lem. 2 and 3 are robust enough to "tolerate" the **WI** stage.

We again partition all the messages in the Q-Q MIM execution of Π_{SSZK} into $L = 24Q^6\lambda$ equal-size blocks, and define the same hybrids G_0 and $\{G_j\}_{j\in[L]}$ as in Sec. 2.4. This time, these blocks and hybrids are defined w.r.t. Π_{SSZK}. That is, in each session, there exist **WI** messages (after the execution of the Prot. 2 instance) that contribute to the total number T of messages in the Q-Q MIM execution. But each block still contain T/L blocks, and the rule to rewind a block does not change (i.e., it is still based on if there are fully nested left gadgets as before).

Additionally, we insert the following hybrid H_{j-1} between G_{j-1} and G_j:[18]

Hybrid H_{j-1} ($\forall j \in [L]$): This hybrid is identical to G_{j-1} (defined on Page 7) except that it additionally monitors the execution of block B_j, and for all left sessions of Π_{SSZK} whose **WI** stage *starts* in block B_j, it switches to using the trapdoor (i.e., more than Th gadgets match) to finish that **WI** stage.

Notice that if we manage to show Lem. 2 and 3 for this new sequence of hybrids (with the intermediate H_j's), the Q-Q simulation soundness of Π_{SSZK} is established. Because in G_L, we are able to use the trapdoor to cheat in the **WI** stage against the adversary in all left sessions (due to Lem. 2); Meanwhile, the adversary cannot use the trapdoor and thus cannot prove false statements in all right sessions (due to the invariant condition[19] claimed in Lem. 3). It turns out the only step that requires new ideas is the switch from G_{j-1} to the new hybrid H_{j-1}. In the following, we focus on the challenges and how we resolve them.

Indistinguishability. We first claim that the view of $\mathcal{A}$ is computationally indistinguishable between G_{j-1} and H_{j-1} ($\forall j \in [L]$). Recall that G_{j-1} performs Watrous rewinding only for the first $j-1$ blocks $\{B_1, \ldots, B_{j-1}\}$. In particular, it means that the WI that *starts* in block B_j will be executed in straight-line. Since the only difference between G_{j-1} and H_{j-1} is the witness used in this WI, the view indistinguishability follows directly from the WI property.

There are two issues to address in the above argument. First, recall that both G_{j-1} and H_{j-1} are coherent executions, so we cannot use the WI property, which is about the de-coherent execution of the **WI** stage. This can be resolved using the same technique as for Lem. 3—We consider an intermediate hybrid G'_{j-1} (resp. H'_{j-1}) that is identical to G_{j-1} (resp. H_{j-1}) but execute the blocks $\{B_j, \ldots, B_L\}$ (i.e., the blocks Watrous rewinding has not reached) de-coherently. In this way, we can perform the reduction to the WI property as explained above.

Second, we need to show that the trapdoor will indeed become available when H_{j-1} needs it. We want to prove this using (a similar argument as for) Lem. 2—When the **WI** stage of some left session *starts* in block B_j, it means the **Preamgle** stage of this left session must complete before block B_j. It then follows from Lem. 2 that the trapdoor witness of this left session must be available. However, the result in Lem. 2 is about G_j that performs Watrous rewinding up to block B_j, but the current hybrid G'_{j-1} only performs rewinding up to B_{j-1}. It is possible that one less gadget is made match in G'_{j-1} (i.e., the one fully nested in block B_j and picked by Watrous' rewinding). Fortunately, this does not affect the availability of trapdoor in G'_{j-1} because the bound in Lem. 2 is derived asymptotically on the security parameter λ, and it still holds even if one less gadget matches. Another related issue is: Compared with the G_j considered in Lem. 2, there are more messages in each session (i.e., the **WI** stage) in G'_{j-1}. But this does not affect the asymptotic bound in Lem. 2 either, because the **WI** stage contribute to each session only a constant number (particularly, independent of λ) of extra messages.

[17] Note that the invariant condition only help us to upper bound the *expected* number of matching right gadgets. But using a proper concentration bound, we can also show that $\mathcal{A}$ cannot make more than Th gadgets match.

[18] That is, the current order of hybrids is: $G_0 \to H_0 \to G_1 \to H_1 \to G_2 \to \cdots \to G_{L-1} \to H_{L-1} \to G_L$.

[19] Note that the invariant condition only help us to upper bound the *expected* number of matching right gadgets. But using a proper concentration bound, we can also show that $\mathcal{A}$ cannot make more than Th gadgets match.

Invariant Condition. We also need to prove the $\mathsf{invariant\ condition}$ (as per Def. 1) in H_{j-1}. A simple solution is to require the **WI** stage to be *statistically* WI. In this way, the switch of witness in H_{j-1} does not affect the $\mathsf{invariant\ condition}$ as the WI execution contains no information of the used witness at all. However, constant-round[20] statistical WI arguments are not unknown from the *minimal* assumption of PQ-OWFs. We thus take a different proof approach (that allows us to keep using the computational WI arguments).

We will keep using the H'_{j-1} defined above, because of its advantage that the blocks after B_{j-1} are de-coherent (again, due to the deferred measurement principle, $\mathsf{invariant\ condition}$ in H'_{j-1} implies that in H_{j-1}). We divide all possible schedules into two types in the same manner as in the proof of Lem. 3 (shown on Page 9). **Type-1** schedules can be handled in exactly the same manner as on Page 9. However, we cannot re-use the same proof for **Type-2** schedules, because the change in the current H'_{j-1} is to switch the WI witness; And the ENMC is non-malleable only w.r.t. another ENMC (but not w.r.t. the concerned WI argument).

Instead, we re-use the proof for G_0 (shown on Page 8) for **Type-2** schedules—If $\mathcal{A}$ manages to break the $\mathsf{invariant\ condition}$, then we can extract the $\widetilde{a}^{(i)}_k$ (due to the extractability of ENMC) and break the computationally hiding property of SBCom using $\widetilde{a}^{(i)}_k$ as a reasonable guess for the value $\widetilde{b}^{(i)}_k$ committed in the corresponding SBCom. However, there is one difficulty in the current setting. **Type-2** schedules in H'_{j-1} only guarantees the ENMC to $\widetilde{a}^{(i)}_k$ starts after the final message of B_{j-1} (and thus is executed de-coherently and in straight-line). However, it is possible that the SBCom to $\widetilde{b}^{(i)}_k$ happens within some block B_z with $z \le j-1$; Since this B_z is performed using Watrous rewinding, we *cannot* view the concerned $\widetilde{\mathsf{gadget}}^{(i)}_k$ as a straight-line, de-coherent execution and perform the above reduction to the hiding property of SBCom.

To solve this issue, we define another hybrid H''_{j-1}, which is identical to H'_{j-1} except that it guess at random the block index z and stop performing the Watrous rewinding for block B_z, while executing other blocks in the same manner as H'_{j-1}. In this way, if the SBCom to $\widetilde{b}^{(i)}_k$ appears within B_z, it will be a de-coherent execution. Moreover, H''_{j-1} is equivalent to H'_{j-1} with probability $1/2$. I.e., they are equivalent as long as the $\mathsf{Watrous\ control\ register}$ $\mathtt{W}_z$ (for the picked z) is set to 0 in hybrid H''_{j-1}, and as we proved earlier, each $\mathsf{Watrous\ control\ register}$ will be set to 0 with probability exactly $1/2$. Then, we can perform the above reduction to the hiding of SBCom in the new hybrid H''_{j-1}. This approach only incurs an $\frac{1}{2} \cdot \frac{1}{\ell_{\mathsf{gad}}}$ multiplicative loss on the adversary's advantage in winning the hiding game, where the term $\frac{1}{2}$ is due to the fact that $H''_{j-1} \equiv H'_{j-1}$ with probability $\frac{1}{2}$, and the term $\frac{1}{\ell_{\mathsf{gad}}}$ is due to the fact that H''_{j-1} needs to guess *correctly* in which block B_z the SBCom to $\widetilde{b}_{i,k}$ will appear. Since ℓ_{gad} is a polynomial of λ, the reduction still works.

Remark 2. It is worth noting the following points regarding the above proof:

1. It is not necessary that the post-quantum non-malleable commitment (PQ-NMC) has to be constant-round. But it is worth noting that [LPY22] is essential to our construction due to its two extra properties: first-message binding and extractability. No other known constructions of PQ-NMCs achieve these two properties simultaneously.

2. In contrast to the PQ-NMC, we do require the **WI** stage to be constant-round. The short explanation is that our proof strategy for simulation soundness scales *exponentially* with the number of rounds of the primitives, and therefore it cannot go beyond constant. (See the paragraph starting with "Simulation Soundness" on Page 28 for details.)

Extension to QMA. Notice that the above security proof for Π_{SSZK} makes use of its **WI** stage in a "black-box" manner. That is, all the claims above hold as long as the **WI** stage is constant-round, computationally WI, and computationally sound. In particular, this is true even if this **WI** stage involves quantum communication. Therefore, a bounded-concurrent simulation-sound ZK argument for **QMA** can be constructed by replacing the **WI** stage for **NP** in Π_{SSZK} with a WI argument for **QMA**, where again the prover proves *either*

[20] If the WI argument is not constant-round, the above proof of indistinguishability may not goes through anymore. Because it is unclear if the asymptotic bound in Lem. 2 still holds.

the **QMA** statement is true *or* the trapdoor statement is true. We remark that such a constant-round WI argument for **QMA** is know from PQ-OWFs in [CCLY22].[21]

2.6 More Applications: Coin-Flipping Protocols, PQ-2PCC, and 2PQC

To build these applications, we want to follow the same template as in Sec. 2.5. That is, we ask the parties to run a **Preamble** stage involving executions of Prot. 2. Then, the parties execute some extra components implementing the desired functionality (e.g., the **WI** stage in our $\Pi_{\textsc{sszk}}$ protocol). As long as the "extra components" are constant-round and have a straight-line security proof, the proof techniques in Sec. 2.5 will generalize to this new construction.

But coin-flipping protocols, or general-purpose 2PC in general, have crucial differences with zero-knowledge arguments:

1. For the Q-Q MIM execution of zero-knowledge arguments, we only need to deal with "fixed-role" corruptions. That is, we know for sure that the MIM adversary corrupts the verifiers of all the left sessions and corrupts the provers of all the right sessions. In contrast, for the Q-concurrent execution of 2PC protocols, it is possible that $\mathcal{A}$ corrupts P_1 of some sessions and P_2 of other sessions (this is typically referred to as *interchangeable-role* corruptions).

2. ZK arguments require *simulation-based* security against corrupted verifiers (i.e., the ZK property), and in this case, the corrupted party (i.e., the verifier) does not have private input. On the other hand, the security requirement against corrupted provers (i.e., the soundness property) is *game-based*. Namely, to prove soundness, the reduction does not need to extract private input (i.e., a potential witness) from a malicious prover. In contrast, for 2PC protocols, we always require a simulation-based security no matter which party is corrupted, and the simulator needs to extract the private input of the corrupted party explicitly. Indeed, this is one of the reasons why ZK arguments are typically easier to build than general-purpose 2PC protocols.

To address Issue 1, we require that P_1 and P_2 in each session execute two sequential instances of Prot. 2 *in opposite directions* in the **Preamble** stage. That is, they first run an instance of Prot. 2 where P_1 acts as the sender S, and then another instance where P_2 acts as the sender S. In the security proof,[22] the schedule of the Q-concurrent execution of the protocol (to be constructed) can be "recast" to a similar pattern as the Q-Q MIM execution of the ZK protocol discussion in Sec. 2.5 with the adversary sitting in the middle. To do that, we simply put Q instances of Prot. 2 *where the receiver is corrupted* on the left, and put the other Q instances of Prot. 2 *where the sender is corrupted* on the right. This matches exactly the Q-Q MIM setting where we proved security of Prot. 2 (and the Q-Q simulation-sound ZK in Sec. 2.5). Intuitively, this allows us to construct a simulator who is able to use the trapdoor in each session when simulating the behavior of honest parties for the MIM adversary $\mathcal{A}$; Meanwhile, $\mathcal{A}$ cannot use any trapdoor. However, to make the the proof work, we have to make sure that the other components (except for the **Preamble** stage) in each session have only constant rounds and their security proof does not involve rewinding. (E.g., for the $\Pi_{\textsc{sszk}}$ in Sec. 2.5, the **WI** stage plays the role of "other components".) We need to instantiate these components carefully to satisfy this requirement (see the following application-specific discussions).

To address Issue 2, our idea is application-specific. In the sequel, we discuss the case of coin-flipping protocols, 2PC for classical functionalities and quantum functionalities one by one.

Two-Party Coin-Flipping Protocols. We start with a canonical construction: Each party P_b ($b \in \{0, 1\}$) sequentially commits to a random share r_b using an *extractable* commitment scheme. Then, they sequentially reveal the committed r_b, *without giving the associated decommitment information.* Finally, they sequentially give a ZK argument to prove that the revealed r_b is indeed the value committed earlier. The coin-flipping result is defined by $r := r_0 \oplus r_1$. To prove the security, a simulator can extract the committed share r_b from the extractable commitment given by the corrupted party, then enforce the coin-flipping result to the r from the ideal functionality by setting $r_{1-b} := r \oplus r_b$ as the (simulated) revealed share and cheating in the ZK argument (using the ZK simulator).

[21] In more detail, [CCLY22] constructed a constant-round ε-ZK argument for **QMA** using only PQ-OWFs. It is well-known that ε-ZK implies WI.

[22] W.l.o.g., we assume that exactly one party is corrupted in each session, in this Q-concurrent execution of 2PCC.

As discussed earlier, if we find a constant-round, straight-line version of the above canonical two-party coin-flipping protocol, and add it after the aforementioned **Preamble** stage, then we will obtain a secure construction re-using the same proof techniques shown in Sec. 2.5:

– For the commitment to share r_b, we ask each party P_b to use a post-quantum statistically binding commitment, and then perform a constant-round *post-quantum secure function evaluation* (SFE) [BD18], so that if the other party P_{1-b} knows the $\mathsf{trapdoor}$,[23] the SFE will reveal the committed value; Otherwise, the SFE reveals nothing.

– Note that after revealing the r_b's, each party also needs to prove their honesty for the execution so far. Instead of using a ZK argument, we ask each party to give a WI argument, where the prover proves that *either* the revealed r_b, the commitment, and the SFE are generated honestly and consistently, *or* the $\mathsf{trapdoor}$ statement is true.

Extension to General-Purpose PQ-2PCC. Bounded concurrent PQ-2PCC can be constructed by taking a (stand-alone-secure) PQ-2PCC protocol Π_{2PCC} *in the CRS model* and using the above bounded-concurrent two-party coin-flipping protocol to generate its CRS. Similar as in the previous constructions, as long as Π_{2PCC} is constant-round and has a straight-line simulator (in the CRS model), the same proof techniques can be re-used to show its security in the bounded-concurrent setting. We remark that such a Π_{2PCC} is known from the quantum hardness of learning with Errors (LWE) (by combining [PVW08] and [GS18]).

Extension to General-Purpose 2PQC. This part is similar to our generalization of Π_{SSZK} from **NP** to **QMA** (shown on Page 11)—The above construction of the PQ-2PCC protocol makes use of the Π_{2PCC} in a rather "black-box" manner. That is, all the claims above hold as long as the Π_{2PCC} is constant-round and straight-line simulatable in the CRS model. In particular, it does *not* rely on the fact that the Π_{2PCC} messages are classical. Therefore, a bounded-concurrent 2PQC can be constructed by replacing the Π_{2PCC} in the above protocol with a constant-round, straight-line simulatable (in the CRS model) 2PC for *quantum functionalities.*

2.7 Generalization to the Multi-Party Setting

The above results for two-party coin-flipping and secure computation for both classical and quantum functionalities generalizes to the multi-party setting, if we setup the parameters of the Prot. 2 instances in the **Preamble** stage carefully.

Let us consider the direct generalization of the two-part protocol in Sec. 2.6 to the n-party case. Compared with the two-party protocol, the main difference is the **Preamble** stage (other stages are less relevant in this overview and thus suppressed). In the two-party setting, the **Preamble** stage consists of two executions of Prot. 2, where in one execution, P_1 plays the roles of the sender S and in the other, P_2 plays the roles of the sender S. In the n-party setting, there will be two executions of Prot. 2 between *every pair of parties* P_i and P_j in the **Preamble** stage.

Here, the key observation is: *The* **Preamble** *stage in the n-party setting can be understood as $\binom{n}{2}$ concurrent executions of two-party* **Preamble Stages**. The analysis of the two-party setting (i.e., Sec. 2.6) shows that if we set the parameter $\ell_{\mathsf{gad}} = 120Q^7\lambda$ in Prot. 2, then the **Preamble** stage will provide "simulation soundness" in the Q-concurrent execution. Therefore, using the above observation, if we set $\ell_{\mathsf{gad}} = 120\widehat{Q}^7\lambda$ with a new $\widehat{Q} \coloneqq \binom{n}{2} \cdot Q$, then the **Preamble** stage will provide "simulation soundness" in the Q-concurrent execution *of the n-party protocol.* Remaining steps of the security proof are almost identical to that for the two-party setting. We refer to Sec. 7 for details.

2.8 Impossibility of (Unbounded) Concurrent 2PQC

We provide a sketch of the impossibility for concurrent 2PC in the quantum settings, where the adversary is allowed to be quantum, and the protocol is computing a quantum functionality. Our approach is based on the classical result of Barak et al. [BPS06], but with important differences that we explain in the following.

[23] Specifically, $\mathsf{trapdoor}$ in this setting is the witness for the following $\mathsf{trapdoor}$ statement: More than $\mathsf{Th} = 60Q^7\lambda + Q^4\lambda$ $\mathsf{gadgets}$ match in the Prot. 2 instance (from the **Preamble** stage) where P_{1-b} acts as the sender S.

To better understand the challenges involved, let us first recall the high-level intuition for the impossibility of concurrent zero-knowledge with respect to an oracle $\mathcal{O}$, described in [BPS06]. Let $\mathit{\Pi}$ be an ℓ-round zero-knowledge protocol to prove the knowledge of a pre-image of a one-way function, the oracle $\mathcal{O}$ plays the role of the verifier, except that in the last round it outputs some secret information, if the (interactive) proof verifies. The idea of the separation consists of the following two steps:

1. An adversary interacting in with an honest prover (the real, concurrent protocol) can trivially recover the secret by simply forwarding the messages of the honest prover to $\mathcal{O}$. Since the honest prover is assumed to succeed, then the adversary can recover the secret with certainty.
2. An adversary interacting with an ideal functionality (the ideal settings) cannot recover the secret, since that would require to complete a proof for a witness that the adversary does not posses.

Although this captures the main idea, the proof is actually more subtle, especially for Step 2. The issue is that the oracle $\mathcal{O}$ is not allowed to keep a state across different queries and therefore the adversary may try to break the zero-knowledge proof by "rewinding" the verifier (represented by the oracle $\mathcal{O}$) or scheduling messages in the wrong order. The way this is solved classically is to let the verifier *sign* the transcript of the protocol in such a way that in the later query it can enforce a straight-line execution of the protocol.

Unfortunately, quantumly this idea does not work, since there is no well-defined notion of "transcript" of quantum protocols, as quantum states cannot in general be copied. The way we solve this issue (which significantly complicates the analysis) is to let the oracle compute the internal state (which could be a quantum state), authenticate it with a quantum authentication code[24], and return it as an output. Then the adversary will be forced to feed the same state in the next query (thus forcing a straight-line execution); Otherwise, the authentication process will fail and the oracle will just return $\perp$. Interestingly, in our proof we need a strong notion of security for quantum authentication codes, namely simulation security. Fortunately for us, the work of Broadbent and Wainewright [BW16] shows that the Clifford code and the trap code satisfy our desired notion of security.

Finally, to lift the theorem from an oracle separation to an actual impossibility, we need to get rid of the oracle $\mathcal{O}$ and substitute it with a two-party functionality, which is not allowed to depend on the protocol $\mathit{\Pi}$. In [BPS06], this is done by substituting the ℓ invocations of the oracle with garbled circuits (which will be given as part of the parties' inputs) and letting the functionality compute the encoding for the inputs. This way, the adversary can *simulate* the calls to $\mathcal{O}$ by simply evaluating the garbled circuits. It turns out that the same idea works in our settings, using recent results on quantum garbled circuits [BY22]. The only subtlety is that we need the garbling scheme to satisfy *adaptive security*, i.e., the distinguisher should be allowed to choose the input adaptively, after it obtains the garbled circuit. To complete the proof, we show that this can be achieved with a simple generic transformation. For more details, we refer the reader to Sec. 8.

3 Preliminaries

Basic Notations. Let $\lambda \in \mathbb{N}$ denote security parameter. For a positive integer n, let $[n]$ denote the set $\{1, 2, ..., n\}$. For a finite set $\mathcal{X}$, $x \xleftarrow{\$} \mathcal{X}$ means that x is uniformly chosen from $\mathcal{X}$. We denote by $\mathsf{poly}(\cdot)$ an unspecified polynomial and by $\mathsf{negl}(\cdot)$ an unspecified negligible function. For two probabilities $p_1(\lambda)$ and $p_2(\lambda)$, we will often use $p_1 = p_2 \pm \mathsf{negl}(\lambda)$ as a shorthand for $|p_1 - p_2| \leq \mathsf{negl}(\lambda)$.

For indistinguishability, we may consider random variables over bit strings or over quantum states. This will be clear from the context. For ensembles of random variables $\mathcal{X} = \{X_i\}_{\lambda \in \mathbb{N}, i \in I_\lambda}$ and $\mathcal{Y} = \{Y_i\}_{\lambda \in \mathbb{N}, i \in I_\lambda}$ over the same set of indices $I = \bigcup_{\lambda \in \mathbb{N}} I_\lambda$ and a function $\varepsilon(\cdot)$, we use $\mathcal{X} \overset{c}{\approx}_\varepsilon \mathcal{Y}$ to mean that for any non-uniform QPT[25] algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $i \in I_\lambda$, we have

$$|\Pr[\mathcal{A}(X_i)] - \Pr[\mathcal{A}(Y_i)]| \leq \varepsilon(\lambda) + \mathsf{negl}(\lambda).$$

[24] Note that a good quantum authentication code also serves as an encryption scheme. Therefore, given this authenticated internal state to the adversary does not reveal information about the verifier's secrets.

[25] Unless stated differently, throughout this paper, computational indistinguishability is always w.r.t. non-uniform QPT adversaries.

We say that $\mathcal{X}$ and $\mathcal{Y}$ are ε-computationally indistinguishable if the above holds. In particular, when the above holds for $\varepsilon = 0$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable, and simply write $\mathcal{X} \overset{\text{c}}{\approx} \mathcal{Y}$. Statistical indistinguishability (denoted by "$\overset{\text{s}}{\approx}_\varepsilon$" and "$\overset{\text{s}}{\approx}$") can be defined similarly but for computationally unbounded adversaries. Moreover, we write $\mathcal{X} \overset{\text{i.d.}}{=\!=} \mathcal{Y}$ to mean that X_i and Y_i are distributed identically for all $i \in I$.

OR-Composition of NP Languages. For an **NP** language $\mathcal{L}$ and a true statement in this language $x \in \mathcal{L}$, we use $\mathcal{R}_\mathcal{L}(x)$ ($\mathcal{R}$ stands for "relation") to denote the set of all witnesses for x. We will refer to the OR-composition of **NP** languages, which is defined in Def. 2.

Definition 2 (OR-Composition of NP Languages). *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two* **NP** *languages. The* OR-composition *of them (dubbed $\mathcal{L}_1 \vee \mathcal{L}_2$) is the new* **NP** *language defined as follows:*

$$\mathcal{L}_1 \vee \mathcal{L}_2 \coloneqq \{(x_1, x_2) \mid x_1 \in \mathcal{L}_1 \vee x_2 \in \mathcal{L}_2\}.$$

3.1 Post-Quantum Commitments and Extractable Commitments

Def. 3 to 7 in this subsection are taken from [CCLY22] with cosmetic modifications customized to our applications.

Definition 3 (Post-Quantum Commitment). *A* post-quantum commitment scheme Π *is a classical interactive protocol between interactive* PPT *machines C and R. Let $m \in \{0,1\}^{\ell(\lambda)}$ (where $\ell(\cdot)$ is some polynomial) is a message that C wants to commit to. The protocol consists of the following stages:*

- **Commit Stage:** *$C(m)$ and R interact with each other to generate a transcript (which is also called a commitment) denoted by* com, *C's state ST_C, and R's output $b_{\mathrm{com}} \in \{0,1\}$ indicating acceptance (i.e., $b_{\mathrm{com}} = 1$) or rejection (i.e., $b_{\mathrm{com}} = 0$). We denote this execution by $(\mathsf{com}, \mathsf{ST}_C, b_{\mathrm{com}}) \leftarrow \langle C(m), R\rangle(1^\lambda)$. When C is honest, ST_C is classical, but when we consider a malicious quantum committer $C^*(\rho)$, we allow it to generate any quantum state ST_{C^*}. Similarly, a malicious quantum receiver $R^*(\rho)$ can output any quantum state, which we denote by OUT_{R^*} instead of b_{com}.*
- **Decommit Stage:** *C generates a decommitment* decom *from ST_C. We denote this procedure by $\mathsf{decom} \leftarrow C(\mathsf{ST}_C)$. Then it sends a message m and decommitment* decom *to R, and R outputs a bit $b_{\mathrm{dec}} \in \{0,1\}$ indicating acceptance (i.e., $b_{\mathrm{dec}} = 1$) or rejection (i.e., $b_{\mathrm{dec}} = 0$). We assume that R's verification procedure is deterministic and denote it by $\mathsf{Verify}(\mathsf{com}, m, \mathsf{decom})$. W.l.o.g., we assume that R always rejects (i.e., $\mathsf{Verify}(\mathsf{com}, \cdot, \cdot) = 0$) whenever $b_{\mathrm{com}} = 0$.*

The scheme satisfies the following correctness requirement:

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

$$\Pr\left[b_{\mathrm{com}} = b_{\mathrm{dec}} = 1 : \begin{array}{l} (\mathsf{com}, \mathsf{ST}_C, b_{\mathrm{com}}) \leftarrow \langle C(m), R\rangle(1^\lambda) \\ \mathsf{decom} \leftarrow C(\mathsf{ST}_C) \\ b_{\mathrm{dec}} \leftarrow \mathsf{Verify}(\mathsf{com}, m, \mathsf{decom}) \end{array}\right] = 1.$$

Definition 4 (Computationally Hiding). *A post-quantum commitment Π is* computationally hiding *if for any $m_0, m_1 \in \{0,1\}^{\ell(\lambda)}$ and any non-uniform QPT receiver $R^*(\rho)$, the following holds:*

$$\{\mathsf{OUT}_{R^*} : (\mathsf{com}, \mathsf{ST}_C, \mathsf{OUT}_{R^*}) \leftarrow \langle C(m_0), R^*(\rho)\rangle(1^\lambda)\}_\lambda \overset{\text{c}}{\approx} \{\mathsf{OUT}_{R^*} : (\mathsf{com}, \mathsf{ST}_C, \mathsf{OUT}_{R^*})\langle C(m_1), R^*(\rho)\rangle(1^\lambda)\}_\lambda.$$

Definition 5 (Statistically Binding). *A post-quantum commitment Π is* statistically binding *if for any unbounded-time comitter C^*, the following holds:*

$$\Pr\left[\begin{array}{l} \exists\ m, m', \mathsf{decom}, \mathsf{decom}',\ \ s.t.\ m \neq m' \wedge \\ \mathsf{Verify}(\mathsf{com}, m, \mathsf{decom}) = \mathsf{Verify}(\mathsf{com}, m', \mathsf{decom}') = 1 \end{array} : (\mathsf{com}, \mathsf{ST}_{C^*}, b_{\mathrm{com}}) \leftarrow \langle C^*, R\rangle(1^\lambda)\right] = \mathsf{negl}(\lambda).$$

Next, we define extractable commitments in the post-quantum setting. We first need a notation for the "committed" value.

Definition 6 (Committed Values). *For a post-quantum commitment* Π*, we define the value function as follows:*

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

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

Definition 7 (ε-Simulatable Extractability). *A commitment scheme* Π *is* extractable with ε-simulation *if there exists a QPT algorithm* $\mathcal{SE}$ *(called the* ε*-simulation extractor) such that for any noticeable* $\varepsilon(\lambda)$ *and any non-uniform QPT* $C^*(\rho)$*,*

$$\left\{\mathcal{SE}^{C^*(\rho)}(1^\lambda, 1^{\varepsilon^{-1}})\right\}_\lambda \overset{s}{\approx}_\varepsilon \left\{(\mathsf{val}_\Pi(\mathsf{com}), \mathsf{ST}_{C^*}) : (\mathsf{com}, \mathsf{ST}_{C^*}, b_{\mathrm{com}}) \leftarrow \langle C^*(\rho), R\rangle(1^\lambda)\right\}_\lambda. \quad (5)$$

Remark 3. Def. 7 is identical to [CCLY22, Definition 11], except that we require *statistical* ε-closeness in Eq. (5) while [CCLY22, Definition 11] only requires *computational* ε-closeness. We emphasis that the security proof in [CCLY22] actually already shows that their construction achieves statistical ε-closeness. Roughly, the reason is their $\mathcal{SE}$ uses identically distributed random coins as the honest receiver so that the simulated execution is indeed identically distributed as the real one (up to the ε simulation error).

3.2 Post-Quantum Non-Malleable Commitments

The following definition of post-quantum non-malleable commitments are taken from [LPY22].

Man-in-the-Middle Execution. Consider a (non-uniform) QPT man-in-the-middle adversary $\mathcal{M} = \{\mathcal{M}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ interacting with a committer C on the *left*, and a receiver R on the *right*. We denote the relevant entities used in the right interaction as the "tilde'd" version of the corresponding entities on the left. In particular, suppose that C commits to m in the left interaction, and $\mathcal{M}$ commits to $\widetilde{m}$ on the right, i.e., we set $\widetilde{m} = \mathsf{val}(\widetilde{\tau})$ where $\widetilde{\tau}$ is the transcript of the right session. Let $\mathsf{mim}^{\mathcal{M}}_{\langle C,R\rangle}(\lambda, m, \rho_\lambda)$ denote the random variable that is the pair $(\mathsf{OUT}_{\mathcal{M}}, \widetilde{m})$, consisting of $\mathcal{M}$'s output as well as the value committed to by $\mathcal{M}$ on the right (assuming C commits to m on the left), where ρ_λ is $\mathcal{M}$'s non-uniform advice. We use an identity-based specification, and ensure that $\mathcal{M}$ uses a distinct ID $\widetilde{\mathsf{id}}$ on the right from the ID id it uses on the left. This is done by stipulating that $\mathsf{mim}^{\mathcal{M}}_{\langle C,R\rangle}(\lambda, m, \rho_\lambda)$ outputs a special value $\bot_{\mathsf{id}}$ when $\mathcal{M}$ uses the same ID in both the left and right executions. The reasoning is that this corresponds to the uninteresting case when $\mathcal{M}$ is simply acting as a channel, forwarding messages from C on the left to R on the right and vice versa.

Definition 8 (Post-Quantum Non-Malleable Commitments). *An identity-based commitment scheme* $\langle C, R\rangle$ *with identity space is said to be* post-quantumly non-malleable *if for every (non-uniform) QPT man-in-the-middle adversary* $\mathcal{M} = \{\mathcal{M}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ *and every polynomial* $\ell : \mathbb{N} \to \mathbb{N}$*, it holds that*

$$\left\{\mathsf{mim}^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(\lambda, m_0, \rho_\lambda)\right\}_{\lambda\in\mathbb{N}, m_0, m_1 \in \{0,1\}^{\ell(\lambda)}} \overset{c}{\approx} \left\{\mathsf{mim}^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(\lambda, m_1, \rho_\lambda)\right\}_{\lambda\in\mathbb{N}, m_0, m_1 \in \{0,1\}^{\ell(\lambda)}}.$$

Lemma 4 ([LPY22]). *Assuming the existence of post-quantum one-way functions, there exists a constant-round construction of post-quantum non-malleable commitments supporting a* $O(2^\lambda)$*-size identity space. Moreover, this construction is also* ε*-simulation extractable (as per Def. 7).*

Remark 4 (First-Message Binding). The [LPY22] post-quantum non-malleable commitments $\langle C, R\rangle$ enjoys the following "first-message binding" property. The first two messages of $\langle C, R\rangle$ are exactly an execution of Naor's commitment, where C (i.e., the committer of the non-malleable commitment) acts as Naor's committer and R (i.e., the receiver of the non-malleable commitment) acts as Naor's receiver, and the message being committed in this Naor's commitment is exactly the message being committed in the non-malleable commitment. Therefore, if we use m to denote the value committed in this Naor's commitment, then the value statistically bound in the non-malleable commitment can be defined as $\mathsf{Val}_d(m) := \begin{cases} m & \text{if } d = 1 \\ \bot & \text{if } d = 0 \end{cases}$, where d denotes R' decision to accept (when $d = 1$) or reject (when $d = 0$) the execution of the non-malleable commitment $\langle C, R\rangle$. This property plays a critical role in our security proof.

Moreover, in our constructions, this non-malleable commitment will be used as a building block in some larger protocol. In that case, this Naor's commitment can be regarded as non-interactive, because the first message of Naor's commitment can be sent in the very beginning of the larger protocol (see, e.g., Stage 1 of Prot. 4). This is why we call this property "first-message binding" (instead of "first-two-messages binding").

3.3 Post-Quantum Secure Function Evaluation

We define secure function evaluation protocols with statistical circuit privacy and quantum input privacy. The following materials are taken from [ABG+21].

Definition 9 (Post-Quantum 2-Message Function-Hiding SFE). *A post-quantum two-message* function-hiding secure function evaluation *(SFE) protocol* SFE *consists of four algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$*. They satisfy the following syntax:*

- $\mathsf{k} \leftarrow \mathsf{Gen}(1^\lambda)$*: It is a PPT algorithm that takes as input a security parameter* 1^λ*, and outputs a secret key* k*;*
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{k}, x)$*: It is a PPT algorithm that takes as input a key* k *a string* x*, and outputs a ciphertext* ct*;*
- $\widehat{\mathsf{ct}} \leftarrow \mathsf{Eval}(C, \mathsf{ct})$*: It is a PPT algorithm that takes as input a classical circuit* C *and ciphertext* ct*, and outputs an evaluated ciphertext* $\widehat{\mathsf{ct}}$*;*
- $\widehat{x} \leftarrow \mathsf{Dec}(\mathsf{k}, \widehat{\mathsf{ct}})$*: It is a deterministic algorithm that takes as input a ciphertext* $\widehat{\mathsf{ct}}$*, and outputs a string* $\widehat{x}$*.*

For any polynomial-size family of classical circuits $\{C_\lambda\}_{\lambda\in\mathbb{N}}$*, the scheme satisfies the following properties:*

1. **Correctness:** *For all* $\lambda \in \mathbb{N}$*,* $x \in \{0,1\}^*$*, and* $C \in \mathcal{C}_\lambda$*, it holds that*
$$\Pr\big[\mathsf{Dec}(\mathsf{k}, \widehat{\mathsf{ct}}) = C(x) : \mathsf{k} \leftarrow \mathsf{Gen}(1^\lambda); \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{k}, x); \widehat{\mathsf{ct}} \leftarrow \mathsf{Eval}(C, \mathsf{ct})\big] = 1.$$
2. **Post-Quantum Input Hiding:** *For all polynomial* $\ell(\lambda)$ *and (non-uniform) QPT adversary* $\mathcal{A} = \{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$*, it holds for two length* $\ell(\lambda)$ *messages* $\{x^0_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $\{x^1_\lambda\}_{\lambda\in\mathbb{N}}$ *that*
$$\Pr\Big[\mathcal{A}_\lambda(\mathsf{ct}; \rho_\lambda) = b : b \xleftarrow{\$} \{0,1\}; \mathsf{k} \leftarrow \mathsf{Gen}(1^\lambda); \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{k}, x^b_\lambda)\Big] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$
3. **Statistical Circuit Privacy:** *There exist (potentially unbounded) algorithms* Sim *(simulator) and* Ext *(extractor) such that*
   - *For all* $x \in \{0,1\}^*$ *and* ct *in the support of* $\mathsf{Enc}(\mathsf{k}, x)$ *(for some* k*), it holds that* $\mathsf{Ext}(\mathsf{ct}) = x$*;* **and**
   - $\{\mathsf{Eval}(C, \mathsf{ct}^*)\}_{\lambda\in\mathbb{N}, C\in\mathcal{C}_\lambda, \mathsf{ct}^*\in\{0,1\}^{\mathsf{poly}(\lambda)}} \overset{s}{\approx} \big\{\mathsf{Sim}\big(C(\mathsf{Ext}(\mathsf{ct}^*))\big)\big\}_{\lambda\in\mathbb{N}, C\in\mathcal{C}_\lambda, \mathsf{ct}^*\in\{0,1\}^{\mathsf{poly}(\lambda)}}$.

We will use the following claim in our analysis, which follows directly from Property 3 in Def. 9.

Lemma 5 (Function Hiding). *Let* $\mathsf{SFE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *be an SEF satisfies Def. 9. Let* $\{\mathsf{ct}^*_\lambda\}_{\lambda\in\mathbb{N}}$ *be any (possibly non-ciphertext)* $\mathsf{poly}(\lambda)$*-length string, and let* $\{C^0_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $\{C^1_\lambda\}_{\lambda\in\mathbb{N}}$ *be two families of circuits such that for all* $\lambda \in \mathbb{N}$*,* C^0_λ *and* C^1_λ *have identical truth tables. Then, it holds that*
$$\{\mathsf{SFE.Eval}(C^0_\lambda, \mathsf{ct}^*_\lambda)\}_{\lambda\in\mathbb{N}} \overset{s}{\approx} \{\mathsf{SFE.Eval}(C^1_\lambda, \mathsf{ct}^*_\lambda)\}_{\lambda\in\mathbb{N}}.$$

Secure function evaluation schemes satisfying Def. 9 are known from the quantum hardness of LWE [OPP14, BD18].

3.4 Post-Quantum Bounded-Concurrent Simualtion-Sound ZK Arguments

For any interactive protocol $\langle P, V\rangle$, we use $\langle \{P_i\}_{i\in[m]}, \{V_i\}_{i\in[m]}\rangle$ to denote the concurrent execution of m instances of $\langle P, V\rangle$, where the i-th instance is the execution of $\langle P, V\rangle$ with P_i acting as the prover and V_i acting as the verifier. When defining soundness, we consider the setting where the provers are corrupted; In this case, we write the execution as $\langle P^*, \{V_i\}_{i\in[m]}\rangle$, meaning that a malicious P^* controls all the provers in the m sessions. Similarly, for properties like zero-knowledge, we can define $\langle \{P_i\}_{i\in[m]}, V^*\rangle$, where the verifiers are corrupted.

Definition 10 (Post-Quantum Bounded-Concurrent Interactive Arguments). *Let $\langle P, V\rangle$ be an interactive protocol between a classical PPT prover P and a classical PPT verifier V. Let $m(\lambda)$ be a polynomial of λ. For any* priori fixed *m, $\langle P, V\rangle$ is a post-quantum m-bounded concurrent interactive argument for an* **NP** *language $\mathcal{L}$ if it satisfies the following requirements:*

1. **(Completeness.)** *For any $(x_1, \ldots, x_m) \in \mathcal{L}^m$ and any $(w_1, \ldots, w_m)$ such that $w_i \in \mathcal{R}_{\mathcal{L}}(x_i)$ for all $i \in [m]$, it holds that:*
$$Pr[\textit{All } V_i\textit{'s accept in } \langle \{P_i(x_i, w_i)\}_{i\in[m]}, \{V_i(x_i)\}_{i\in[m]}\rangle] = 1.$$
2. **(Computational Soundness.)** *For any (non-uniform) QPT prover $P^* = \{P^*_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ and any $(x_1, \ldots, x_m) \in (\{0,1\}^\lambda)^m$, it holds that*
$$Pr[\exists i \text{ s.t. } (x_i \notin \mathcal{L}) \wedge (d_i = 1) : (d_1, \ldots, d_m) \leftarrow \mathsf{OUT}_V\big(\langle P^*_\lambda(x_1, \ldots, x_m; \rho_\lambda), \{V_i(x_i)\}_{i\in[m]}\rangle\big)] = \mathsf{negl}(\lambda),$$
where d_i denotes the output of V_i in the execution $\langle P^*_\lambda(x_1, \ldots, x_m; \rho_\lambda), \{V_i(x_i)\}_{i\in[m]}\rangle$, indicating if V_i accepts (when $d_i = 1$) or rejects (when $d_i = 0$) in the i-th session.

Man-in-the-Middle Execution. Let $m(\lambda)$ be a polynomial of λ. Let $\langle P, V\rangle$ be an interactive argument system for a language $\mathcal{L} \in$ **NP** with witness relation $\mathcal{R}_{\mathcal{L}}$. For any (non-uniform) QPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$, consider the following m-m *man-in-the-middle* (MIM) setting:

– m Left Sessions: $\mathcal{A}_\lambda$, on input $(\{x_1, \ldots, x_m\}; \rho_\lambda)$, executes m instances of $\langle P, V\rangle$ with m honest provers $\{P_i(x_i, w_i)\}_{i\in[m]}$. $\mathcal{A}_\lambda$ plays the role of the verifier in all of these m sessions. Similar as for non-malleable commitments, each session is associated with an identity (or tag). We use id_i to denote the identity of the i-th left session.

– m Right Sessions: The same $\mathcal{A}_\lambda$ with the same input executes anther m instances of $\langle P, V\rangle$ with m honest verifiers $\{V_i\}_{i\in[m]}$. $\mathcal{A}_\lambda$ plays the role of the (potentially malicious) provers in all of these m sessions, trying to prove a statement $\widetilde{x}_i$ in the i-th right session. We use $\widetilde{\mathsf{id}}_i$ to denote the identity of the i-th right session.

We emphasize that $\mathcal{A}_\lambda$ participates in the above executions *simultaneously*, taking full control over the schedule of messages on both sides; The statements $\{x_1, \ldots, x_m\}$ proven in the left sessions are given to the corresponding P_i's and $\mathcal{A}_\lambda$ prior to the experiment; In contrast, the statements $\{\widetilde{x}_1, \ldots, \widetilde{x}_m\}$ proven in the right interaction the identities used on both sides are chosen by $\mathcal{A}_\lambda$ during the experiment. Let $\mathsf{View}_{\mathcal{A}_\lambda}(\rho_\lambda, x_1, \ldots, x_m)$ denote the view of $\mathcal{A}_\lambda$ in the above experiment.

Definition 11 (Post-Quantum Bounded-Concurrently Simulation-Sound ZK). *Let $m(\lambda)$ be a polynomial of λ. For any* priori fixed *m, an m-concurrent post-quantum interactive argument system $\langle P, V\rangle$ for an* **NP** *language $\mathcal{L}$ (as per Def. 10) is* post-quantum m-concurrent simulation-sound *if there exists a QPT machine $\mathcal{S}$ (simulator) such that for any (potentially non-uniform) QPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$, the following hold:*

1. **(Indistinguishable Simulation.)** *It holds that*
$$\big\{\mathsf{View}_{\mathcal{A}_\lambda}(\rho_\lambda, x_1, \ldots, x_m)\big\}_{\lambda\in\mathbb{N},\ x_1,\ldots,x_m\in\mathcal{L}\cap\{0,1\}^\lambda} \overset{c}{\approx} \big\{\mathcal{S}(1^\lambda, \rho_\lambda, x_1, \ldots, x_m)\big\}_{\lambda\in\mathbb{N},\ x_1,\ldots,x_m\in\mathcal{L}\cap\{0,1\}^\lambda}.$$
2. **(Simulation Soundness.)** *For any $x_1, \ldots, x_m \in \mathcal{L} \cap \{0,1\}^\lambda$ and any left-session IDs $\{\mathsf{id}_j\}_{j\in[m]}$, it holds that*
$$\Pr\left[\begin{array}{c}\exists i \in [m] \text{ s.t. } \big(\widetilde{x}_i \notin \mathcal{L}\big) \ \wedge\ \big(\widetilde{d}_i = 1\big) \\ \wedge\ \big(\forall j \in [m],\ \mathsf{id}_j \neq \widetilde{\mathsf{id}}_i\big)\end{array} : \mathsf{View} \leftarrow \mathcal{S}(1^\lambda, \rho_\lambda, x_1, \ldots, x_m)\right] \leq \mathsf{negl}(\lambda),$$
where $\{\widetilde{\mathsf{id}}_i\}_{i\in[m]}$ are the IDs of the right sessions as specified in View*, $\widetilde{x}_i$ is the statement in the i-th right session as specified in* View*, and $\widetilde{d}_i$ is the verifier's decision bit in the i-the right session as specified in* View*, indicating if it accepts $(\widetilde{d}_i = 1)$ or rejects $(\widetilde{d}_i = 0)$ this execution.*

Remark 5 (On m-Concurrent ZK). We remark that m-concurrent simulation soundness (as per Def. 11) implies m-concurrent ZK. Thus, we do not need to define bounded-concurrent zero-knowledge separately. The interested reader can find the definition of post-quantum bounded-concurrent ZK in, e.g., [ACL21, Section 3.1].

3.5 Bounded-Concurrent Multi-Party Secure Computation

We refer to [Pas04, Section 2] for the formal model of secure multi-party computation in the bounded-concurrent setting. In the following, we only hight-light the difference due to quantum computation.

In this work we consider a malicious, static adversary, who is a non-uniform QPT machine. That is, at the beginning of the execution the adversary is given a set I of corrupted parties which she controls and through she will not change the set I. Similar as in [Pas04], I could be an *arbitrary* subsets of parties concurrently execute the protocol (dubbed *sessions*), possibly with *interchangeable roles*, meaning that a corrupted party could play the role of party i in one session, but play the role of party $j \neq i$ in another session. The focus of this work is not on fairness. We therefore present a definition where the adversary always receives its own output and can then decide when the honest parties will receive their output. The scheduling of message delivery is decided by the adversary.

Let n denote the total number of parties. In this model, there are n input-selecting machines $M_1, \ldots, M_n$ select input for each party respectively, they are to account for the possibility that the input to parties may depend on previous executions of other protocols in the bounded-concurrent setting (see [Pas04, Section 2] for details). The initial input to the input-selecting machines are defined to be $\overline{x} = (x_1, \ldots, x_n)$. We will also consider computation for quantum functionalities. In that case, the initial input $(x_1, \ldots, x_n)$ could be quantum states.

Parties communicate via authenticated point-to-point channels as well as broadcast channels, where everyone can send messages in the same round. The network is assumed to be synchronous with rushing adversaries, i.e. adversaries may generate their messages for any round after observing the messages of all honest parties in that round, but before observing the messages of honest parties in the next round.

Let f be a (potentially quantum) functionality. The ideal execution of f with security parameter λ, input-selecting machines $M = (M_1, ..., M_n)$, initial inputs $\overline{x} = (x_1, \ldots, x_n)$ and auxiliary input ρ_λ to the ideal-world simulator $\mathcal{S}$, is denoted by $\mathrm{IDEAL}_{f,I,\mathcal{S},M}(\lambda, \overline{x}, \rho_\lambda)$. For a protocol Π, the real-world execution of m-bounded concurrent instances of Π with λ, $M = (M_1, \ldots, M_n)$, $\overline{x} = (x_1, \ldots, x_n)$ and a non-uniform QPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda, \rho_\lambda\}_\lambda$ is denoted by $\mathrm{REAL}^m_{\Pi,I,\mathcal{A},M}(\lambda, \overline{x}, \rho_\lambda)$. We refer to [Pas04, Section 2] for a formal description of the real-world and ideal-world execution.

Definition 12 (Bounded-Concurrent Secure Multi-Party Computation). *Let $m = m(\lambda)$ be a polynomial of the security parameter λ, and let f be a classical (resp. quantum functionality, represented by some quantum circuit). A n-party protocol protocol Π is said to t-securely compute f under m-concurrent composition if for every real-world non-uniform QPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$, there exists an ideal-world non-uniform QPT machine $\mathcal{S}$ (dubbed the simulator), such that for all input-selecting machines $M = M_1, ..., M_n$, every $\overline{x} = (x_1, \ldots, x_n)$, and every $I \subset [n]$ with $|I| < t$, it holds that*

$$\left\{\mathrm{IDEAL}^m_{f,I,\mathcal{S},M}(\lambda, \overline{x}, \rho_\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathrm{REAL}^m_{\Pi,I,\mathcal{A},M}(\lambda, \overline{x}, \rho_\lambda)\right\}_{\lambda\in\mathbb{N}}.$$

That is, m concurrent executions of Π with $\mathcal{A}$ cannot be distinguished from m concurrent invocations of f with S in the ideal world. Moreover, if the honest parties only need to perform classical computation to run Π, we say Π is post-quantum.

3.6 Technical Lemmas

3.6.1 Watrous' Rewinding Lemma

The following is Watrous' rewinding lemma [Wat06] in the form of [ACL21, Lemma 8] with cosmetic modifications.

Lemma 6 (Watrous' Rewinding Lemma [Wat06]). *Let U be a quantum circuit acting on $1 + n + k$ registers such that for every n-qubit state $|\psi\rangle$, the following holds:*

$$U |0\rangle_{\mathtt{W}} |\psi\rangle |0^k\rangle = \sqrt{p(\psi)}\, |0\rangle_{\mathtt{W}} |\phi_0(\psi)\rangle + \sqrt{1 - p(\psi)}\, |1\rangle_{\mathtt{W}} |\phi_1(\psi)\rangle\,,$$

where we will refer to register W *as the* Watrous control register.

Let $p_0, p_1 \in (0, 1)$ and $\varepsilon \in (0, 1/2)$ be real numbers such that:

– $|p(\psi) - p_1| \leq \varepsilon$,

– $p_0(1-p_0) \leq p_1(1-p_1)$, **and**

– $p_0 \leq p(\psi)$,

for all n-qubit input state $|\psi\rangle$. Then, there exists a general quantum circuit W of size $O\big(\frac{\log(1/\varepsilon)\mathsf{size}(U)}{p_0(1-p_0)}\big)$ that on any n-qubit input $|\psi\rangle$ outputs a state $\rho(\psi)$ over $n+k$ qubits satisfying the following property:

$$\langle\phi_0(\psi)|\, \rho(\psi)\, |\phi_0(\psi)\rangle \geq 1 - \frac{16\varepsilon \log^2(1/\varepsilon)}{p_0^2(1-p_0)^2}.$$

In this case, we refer to W as $\mathsf{Amplifier}(U,\varepsilon)$. If ε is a negligible function in the security parameter, we omit this from the algorithm.

3.6.2 Martingales and Azuma–Hoeffding Inequality

In our security proof, we need to establish concentration bounds for some (slightly) dependent random variables. Notice that the Chernoff bound is not applicable in this setting. For our specific application, it turns out that we can define a *martingale* over the concerned random variables and use Azuma–Hoeffding inequality to show a "Chernoff-like" concentration. In the following, we recall the related materials. More details can be found at [MU05, Chapter 13].

Definition 13 (Martingale). *A sequence of random variables $\{Z_0, Z_1, \ldots\}$ is a martingale with respect to the sequence $\{X_0, X_1, \ldots\}$ if, for all $n \geq 0$, the following conditions hold:*

– *Z_n is a function of $\{X_0, X_1, \ldots, X_n\}$;*

– *$\mathbb{E}[|Z_n|] < \infty$;*

– *$\mathbb{E}[Z_{n+1} \mid X_0, \ldots, X_n] = Z_n$.*

Definition 14 (Doob's Martingale). *Let Y be a random variable with $\mathbb{E}[|Y|] < \infty$. Let $\{X_0, X_1, \ldots, X_n\}$ be a sequence of random variables. Then $Z_i \coloneqq \mathbb{E}[Y \mid X_0, \ldots, X_i]$ ($i = 0, 1, ..., n$,) is a martingale, which is called* Doob's martingale*.*

Lemma 7 (Azuma–Hoeffding Inequality). *Let $\{X_0, \ldots, X_n\}$ be a martingale such that*

$$\forall k \in [n],\ |X_k - X_{k-1}| \leq c_k.$$

Then, for all $t \geq 1$ and all $\varepsilon > 0$, it holds that

$$\Pr\left[|X_t - X_0| \geq \varepsilon\right] \leq \frac{2}{e^{\varepsilon^2/(2\sum_{k=1}^t c_k^2)}}.$$

4 Bounded-Concurrent Simulation-Sound ZK Arguments for NP and QMA

4.1 Construction for NP

Our construction for **NP** makes use of the following building blocks:

– A constant-round statistically binding and computationally hiding commitment SBCom. Let $\Gamma_{\text{SBC.C}}$ denote the round complexity of its Commit Phase, and $\Gamma_{\text{SBC.D}}$ the round complexity of its Decommit Phase. Let $\Gamma_{\text{SBC}} \coloneqq \Gamma_{\text{SBC.C}} + \Gamma_{\text{SBC.D}}$. We use Naor's commitment as SBCom.

– A constant-round post-quantum non-malleable commitment ENMC (as per Def. 8) that is also ε-simulation extractable (as per Def. 7) and first-message binding (as per Rmk. 4). Let Γ_{ENMC} denote its round complexity. We use the construction from Lem. 4 as our ENMC.

– A constant-round post-quantum witness indistinguishable argument WI. Let Γ_{WI} denote its round complexity. It is known that such a WI can be obtained assuming only post-quantum OWFs.

Protocol 4: Post-Quantum Bounded-Concurrent Simulation-Sound Zero-Knowledge Argument

Let $Q(\lambda)$ be a polynomial of λ, denoting the maximum number of concurrent sessions. Prover P takes as input λ, $x \in L$, and $w \in \mathcal{R}_{\mathcal{L}}(x)$; Verifier V takes as input λ and $x \in \mathcal{L}$. P and V agree on an $\mathsf{id} \in \{0,1\}^\lambda$ as the ID for this execution of the protocol.

1. In this stage, P (resp. V) samples a random string $\beta_P \xleftarrow{\$} \{0,1\}^\lambda$ (resp. $\beta_V \xleftarrow{\$} \{0,1\}^\lambda$) sends it to V (resp. P). The purpose of β_P and β_V is as follows: In later steps, both P and V need to perform Naor's commitments for several times (recall that Naor's commitment remains secure even if polynomially-many instances use the same first-round message). We stipulate that whenever P (resp. V) needs to make a Naor's commitment to some value, it uses β_V (resp. β_P) as the first message of the two-round Naor's commitment. In this way, we can assume w.l.o.g. that all the Naor's commitments in our construction are non-interactive (i.e., single-round).
2. **(Com-and-Guess Stage.)** For $k = 1$ to $\ell_{\mathsf{gad}} \coloneqq 120Q^7\lambda$:
   (a) V samples $b_k \xleftarrow{\$} \{0,1\}$ and commits to it using SBCom. Note that this is a non-interactive execution of Naor's commitment, as we explained in Stage 1.
   (b) P samples $a_k \xleftarrow{\$} \{0,1\}$ and commits to it using ENMC. Note that the ENMC requires an ID (see Def. 8). P and V will use $\mathsf{id}{:}k$ (according to some standard encoding) as the ID for this ENMC so that each ENMC is executed using a different ID. (Recall that id is the ID for this SSZK.) Also recall that the first two messages of ENMC constitute a Naor's commitment. As discussed in Stage 1, one can think this Naor's commitment as being non-interactive; We denote this message as $\mathsf{com}_k = \mathsf{Com}_{\beta_V}(a_k; r_k)$.
   (c) V sends b_k together with the decommitment information w.r.t. the SBCom in Step 2a. P continues only if this decommitment is valid.

   Terminology: We call each repetition of Step 2a to Step 2c a gadget. Naturally, we call the ℓ_{gad} defined above the $\mathsf{total\ number\ of\ gadgets}$. The k-th gadget *matches* if the a_k committed by P in Step 2b is equal to the b_k *validly* decommitted by V in Step 2c. We emphasize that this $\mathsf{matching}$ condition is always well-defined regardless of the honesty of P (or V), because both SBCom and ENMC are *statistically* binding.
3. **(WI Stage.)** P and V execute an instance of WI where P proves that *either* x is in the language *or* no less than $\mathsf{Th} \coloneqq 60Q^7\lambda + Q^4\lambda$ gadgets are matching. Formally, P proves that $\big(x, \{(\beta_V, \mathsf{com}_k, b_k)\}_{k=1}^{\ell_{\mathsf{gad}}}\big) \in \mathcal{L} \vee \mathcal{L}_{\mathsf{match}}^{\ell_{\mathsf{gad}},\mathsf{Th}}$ (see Def. 2), where

$$\mathcal{L}_{\mathsf{match}}^{\ell_{\mathsf{gad}},\mathsf{Th}} \coloneqq \left\{ \{(\beta_V, \mathsf{com}_k, b_k)\}_{j=1}^{\ell_{\mathsf{gad}}} : \begin{array}{l} \exists G \subseteq [\ell_{\mathsf{gad}}],\ \exists \{(a_k, r_k)\}_{k \in G} \text{ s.t. } |G| \geq \mathsf{Th} \wedge \\ \forall k \in G,\ a_k = b_k \ \wedge\ \mathsf{com}_k = \mathsf{Com}_{\beta_V}(a_k; r_k) \end{array} \right\}, \tag{6}$$

   where by our parameter setting $\ell_{\mathsf{gad}} = 120Q^7\lambda$ and $\mathsf{Th} = 60Q^7\lambda + Q^4\lambda$.

   Terminology: Later in the security proof, we will refer to $w \in \mathcal{R}_{\mathcal{L}}(x)$ as the real witness, and refer to the witness for $\{(\beta_V, \mathsf{com}_k, b_k)\}_{k=1}^{\ell_{\mathsf{gad}}} \in \mathcal{L}_{\mathsf{match}}^{\ell_{\mathsf{gad}},\mathsf{Th}}$ as the $\mathsf{trapdoor}$ witness. Of course, the $\mathsf{trapdoor}$ witness should not exist even for a cheating prover, which we will show when proving soundness. But our ZK simulator will set up the $\mathsf{trapdoor}$ witness and make use of it. Also, we emphasize that the honest P always finishes this stage using the real witness.

Remark 6 (On the IDs). In the concurrent setting, each executing of Prot. 4 uses a different ID. We will consider Q concurrent executions of Prot. 4 where Q is a polynomial of λ. Each instance invokes ℓ_{gad} ENMCs (one for each gadget). Therefore, there will be $Q \cdot \ell_{\mathsf{gad}}$ ENMCs (or $2Q \cdot \ell_{\mathsf{gad}}$ in the Q-Q MIM setting). Recall from Lem. 4 that the ENMC supports a exponential-size identity space. Thus, with our encoding $\mathsf{id}_i{:}k$ (i.e., the k-th ENMC of session id_i uses ID $\mathsf{id}_i{:}k$), all these polynomially many ENMCs use different IDs.

Security Proof. The security of Prot. 4 is established by the following Thm. 3. To prove Thm. 3, first note that completeness follows straightforwardly from the description in Prot. 4. Thus, we only need to establish the Q-concurrent soundness (to show that it is a Q-concurrent interactive argument system) and Q-concurrent simulation soundness, which are provided in Sec. 4.2 and Sec. 4.3 respectively.

Theorem 3. *For any $Q(\lambda)$ that is a polynomial of the security parameter λ, Prot. 4 is a Q-concurrent simulation-sound ZK argument (as per Def. 11).*

4.2 Proving Thm. 3: Q-concurrent Soundness

In this subsection, we prove Q-concurrent soundness of Prot. 4. That is, a (potentially malicious) QPT prover P^* is running Q instances of Prot. 4 concurrently. We want to prove that P^* cannot make V accept a false statement in any session.

We assume for contradiction that for some $i \in [Q]$, P^* makes V_i (the verifier in the i-th session) accept a false statement $x_i \notin \mathcal{L}$ with non-negligible probability, and drive the desired contradiction by constructing a malicious prover P^*_{sd} that breaks the soundness of the Stage 3 WI of the i-th session. In the following, we assume that our reduction knows this session index i. This is without loss of generality because the reduction can always guess i correctly with probability $1/Q$, which will only introduce a $1/Q$ multiplicative factor to P^*_{sd}'s advantage in breaking soundness. This will not affect the validness of our reduction as $1/Q$ is an inverse polynomial of λ.

The P^*_{sd} works as follows:

- It internally emulates the execution between P^* and $\{V_1, \ldots, V_Q\}$ until the beginning of the Stage 3 WI of the i-th session.
- It then relays the messages of the Stage 3 WI of the i-th session, between the internal P^* and the external verifier V_{sd} of the soundness game. All other messages exchanged between P^* and $\{V_1, \ldots, V_Q\}$ are still emulated internally by P^*_{sd}.

If we can prove that the $\mathsf{trapdoor}$ witness is not available to the internal P^* in the i-th session, then it is easy to see that if the internally emulated P^* manages to prove a false statement $x_i \notin \mathcal{L}$ in session i, P^*_{sd} will convince the external V_{sd} on x_i, breaking the soundness of WI. In the following, we prove that the $\mathsf{trapdoor}$ witness is not available session i.

The intuition. At a high-level, our proof proceeds in two steps:

1. First, notice that any gadget of session i matches with probability $\frac{1}{2} \pm \mathsf{negl}(\lambda)$. This is due to the computationally hiding property of SBCom and the extractability of ENMC. In more detail, if P^* manages to make the k-th gadget of session i match with probability non-negligibly greater than half, we can break the computationally hiding property of SBCom to $b_{i,k}$[26] by extracting the value $a_{i,k}$ from the ENMC in the k-th gadget of session i and using this extracted $a_{i,k}$ as a reasonable guess for the $b_{i,k}$ hidden in SBCom.
2. Next, notice that the first bullet implies that in expectation, there will be $\ell_{\mathsf{gad}}/2 \pm \mathsf{negl}(\lambda) = 60Q^7\lambda \pm \mathsf{negl}(\lambda)$ matching $\mathsf{gadgets}$ in session i. Then, it follows from Chernoff bound that except for negligible probability, the total number of matching $\mathsf{gadgets}$ in session i will not reach the threshold $\mathsf{Th} = 60Q^7\lambda + Q^4\lambda$, which implies the unavailability of $\mathsf{trapdoor}$ witness in session i.

Note that the ENMC we are using for Step 1 has an ε statistical simulation error (see Def. 7). But since the reduction has the freedom to set ε to any arbitrarily small inverse polynomial of λ, it will not be a problem. In more detail, when performing the argument for Step 1, we will set $\varepsilon = \frac{\delta(\lambda)}{2}$, where $\delta(\lambda)$ is such that the malicious P^* manages to make some k-th gadget of some i-th session match with probability $\geq \frac{1}{2} + \delta(\lambda)$. Then, the P^*_{sd} we construct from P^* will has advantage $\geq \frac{1}{2} + \delta(\lambda) - \varepsilon = \frac{1}{2} + \frac{\delta(\lambda)}{2}$ in breaking the hiding property of SBCom to $b_{i,k}$. If $\delta(\lambda)$ is non-negligible, $\frac{\delta(\lambda)}{2}$ is also non-negligible. Since this proof is standard, we omit the details.

However, there is a technical issue regarding Step 2—Chernoff bound is *not* applicable. The reason is: Whether a later gadget matches or not could potentially depend on earlier $\mathsf{gadgets}$. Though we know the expectation of the total number of matching $\mathsf{gadgets}$, the status of being matching for different $\mathsf{gadgets}$ is *not independent* random variables, and thus Chernoff bound does not apply. To address this issue, we claim that

[26] This is the value committed in the SBCom of the k-th gadget of the i-th session. Henceforth, we take the convention that the first index in the subscript indicates the session and the second indicates the gadget.

a "Chernoff-like" concentration bound can be established using Doob's martingale and Azuma-Hoeffding inequality as shown in Sec. 3.6.2. In the following, we present a formal treatment.

Formal Proof of Step 2. In this proof, it suffices to focus on an (arbitrary) session $i \in [Q]$. We first define a sequence of random variables $\{X_1, \ldots, X_{\ell_{\mathsf{gad}}}\}$:

– X_k (for $k \in [\ell_{\mathsf{gad}}]$): this is a binary random variable that is equal to 1 iff the k-th gadget of session i matches. (Since we focus on a fixed session i, we do not put i in the subscript of X_k anymore.)

What we have proven in Step 1 can be formally written as:

$$\forall k \in [\ell_{\mathsf{gad}}],\ \ \Pr[X_k] = \frac{1}{2} \pm \mathsf{negl}(\lambda), \tag{7}$$

which further implies:

$$\mathbb{E}[\textstyle\sum_{k=1}^{\ell_{\mathsf{gad}}} X_k] = \frac{1}{2} \cdot \ell_{\mathsf{gad}} \pm \mathsf{negl}(\lambda) = 60Q^7\lambda \pm \mathsf{negl}(\lambda). \tag{8}$$

Recall that our goal in this step is to establish the following inequality:

$$\Pr\Big[\textstyle\sum_{k=1}^{\ell_{\mathsf{gad}}} X_k \geq 60Q^7 + Q^4\lambda\Big] \leq \mathsf{negl}(\lambda). \tag{9}$$

We first define a new sequence of random variables $\{Y_0, \ldots, Y_{\ell_{\mathsf{gad}}}\}$ where

$$Y_0 \coloneqq \mathbb{E}\big[\textstyle\sum_{j \in [\ell_{\mathsf{gad}}]} X_j\big], \ \text{ and } \ \forall k \in [\ell_{\mathsf{gad}}],\ Y_k \coloneqq \mathbb{E}\big[\textstyle\sum_{j \in [\ell_{\mathsf{gad}}]} X_j \mid X_1, \ldots, X_k\big].$$

It follows from the above definition that

$$Y_{\ell_{\mathsf{gad}}} = \mathbb{E}[\textstyle\sum_{j \in [\ell_{\mathsf{gad}}]} X_j \mid X_1, \ldots, X_{\ell_{\mathsf{gad}}}] = \sum_{j \in [\ell_{\mathsf{gad}}]} X_j.$$

Also, since $\sum_{j \in [\ell_{\mathsf{gad}}]} X_j$ has a bounded expectation, this sequence of $\{Y_0, \ldots, Y_{\ell_{\mathsf{gad}}}\}$ forms a Doob's martingale (recall it from Def. 14).

Next, observe that for any $k \in [\ell_{\mathsf{gad}}]$, it holds that (where we use S_k as a shorthand for the sequence $\{X_1, \ldots, X_k\}$)

$$\begin{aligned}
&|Y_k - Y_{k-1}| \\
&= \Big|\mathbb{E}\big[\textstyle\sum_{j \in [\ell_{\mathsf{gad}}]} X_j \mid S_k\big] - \mathbb{E}\big[\textstyle\sum_{j \in [\ell_{\mathsf{gad}}]} X_j \mid S_{k-1}\big]\Big| \\
&= \Big|\textstyle\sum_{j \in [\ell_{\mathsf{gad}}]} \mathbb{E}\big[X_j \mid S_k\big] - \textstyle\sum_{j \in [\ell_{\mathsf{gad}}]} \mathbb{E}\big[X_j \mid S_{k-1}\big]\Big| \\
&\leq \textstyle\sum_{j \in [\ell_{\mathsf{gad}}]} \Big|\mathbb{E}\big[X_j \mid S_k\big] - \mathbb{E}\big[X_j \mid S_{k-1}\big]\Big| \\
&= \sup_{x_1, \ldots, x_k} \Big|\mathbb{E}\big[X_k \mid S_k = (x_1, \ldots, x_k)\big] - \mathbb{E}\big[X_k \mid S_{k-1} = (x_1, \ldots, x_{k-1})\big]\Big| + \sum_{j=k+1}^{\ell_{\mathsf{gad}}} \Big|\mathbb{E}\big[X_j \mid S_k\big] - \mathbb{E}\big[X_j \mid S_{k-1}\big]\Big| \\
&\leq 1 + \sum_{j=k+1}^{\ell_{\mathsf{gad}}} \Big|\mathbb{E}\big[X_j \mid S_k\big] - \mathbb{E}\big[X_j \mid S_{k-1}\big]\Big| && (10) \\
&\leq 1 + (\ell_{\mathsf{gad}} - k) \cdot \mathsf{negl}(\lambda) && (11) \\
&\leq 2 && (12)
\end{aligned}$$

where Eq. (10) follows from the fact that X_j is a *binary* random variable, and Eq. (11) follows from the Claim 4 that we will prove shortly.

Therefore, Eq. (9) follows by applying Azuma-Hoeffding inequality (Lem. 7), with $c_k = 2$ ($\forall k \in [\ell_{\mathsf{gad}}]$) and $\varepsilon = Q^4\lambda$, to bound $|Y_0 - Y_{\ell_{\mathsf{gad}}}|$.

Claim 4. *Let $\{X_1, \ldots, X_{\ell_{\mathsf{gad}}}\}$ and S_k be defined as the above. Assume* SBCom *is computationally hiding and* ENMC *is ε-simulation extractable. Then, it holds that*

$$\forall k \in [\ell_{\mathsf{gad}}],\ \forall j \in \{k+1, \ldots, \ell_{\mathsf{gad}}\},\ \Big|\mathbb{E}\big[X_j \mid S_k\big] - \mathbb{E}\big[X_j \mid S_{k-1}\big]\Big| = \mathsf{negl}(\lambda).$$

Proof. To prove Claim 4, it suffices to show the following stronger claim:

$$\forall k \in [\ell_{\mathsf{gad}}],\ \forall j \in \{k+1, \ldots, \ell_{\mathsf{gad}}\},\ \begin{cases} \Pr[X_j = 1 \mid X_1, \ldots, X_k] = \frac{1}{2} \pm \mathsf{negl}(\lambda) \\ \Pr[X_j = 1 \mid X_1, \ldots, X_{k-1}] = \frac{1}{2} \pm \mathsf{negl}(\lambda) \end{cases}. \tag{13}$$

Recall that we have already proven a similar statement in Step 1 (i.e., Eq. (7)). The only different between Eq. (13) and Eq. (7) is that Eq. (7) is not conditioned on whether earlier $\mathsf{gadgets}$ matches or not (i.e., $X_1, \ldots, X_k$). But it is straightforward to see that Eq. (13) can be proven using the same argument as explained in Step 1. The reason is because when proving Eq. (13) for a particular $j \geq k+1$, all the earlier $\mathsf{gadgets}$ (i.e., $X_1, \ldots, X_k$) can be fixed as non-uniform advice when applying the reduction explained in Step 1.

This finishes the proof of Claim 4. □

4.3 Proving Thm. 3: Q-Concurrent Simulation Soundness

We prove simulation soundness of Prot. 4 in the *Q-concurrent setting*, for any $Q(\lambda)$ that is a polynomial of λ. Recall from Def. 11 that this setting refers to the Q-Q man-in-the-middle execution of Prot. 4.

To prove simulation soundness, we need to construct a simulator $\mathcal{S}$ satisfying the requirements in Def. 11. We will do that through a sequence of hybrids REAL and $\{(H_{j,1}, H_{j,2})\}_{j \in [L]}$. Looking ahead, REAL is the real MIM execution and $H_{L,2}$ will be our simulator where the witnesses $\{w_i\}_{i \in [Q]}$ are not used in the Q left sessions any more.

Notation. Before presenting the hybrids, we first set up some notations. Recall that there are

$$T = 2Q \cdot \big(\ell_{\mathsf{gad}} \cdot (\mathit{\Gamma}_{\mathrm{SBC}} + \mathit{\Gamma}_{\mathrm{ENMC}}) + \mathit{\Gamma}_{\mathrm{WI}}\big)$$

messages in total in the Q-Q MIM execution of Prot. 4. We will partition these T messages into $L \coloneqq 24Q^6\lambda$ equal-size blocks $\{B_1, \ldots, B_L\}$. That is, block B_1 contains the first $\frac{T}{L}$ messages, block B_2 contains the next $\frac{T}{L}$ messages, and so on (messages are ordered according to their order of appearing in the execution). We will refer to L as the *total number* of blocks and refer to $s_B \coloneqq \frac{T}{L}$ as the *size* of each block.

Looking ahead, we need to prove that the $\mathsf{trapdoor}$ witness is unavailable in any of the Q right sessions, in order to argue that $\mathcal{A}$ cannot give a convincing proof for a false $\widetilde{x}_i$ for some $i \in [Q]$. Toward that, we need to define an $\mathsf{invariant\ condition}$ which essentially implies the unavailability of the $\mathsf{trapdoor}$ witness.

Definition 15 (Invariant Condition). *Let $\mathcal{T}$ be a transcript corresponding to an Q-Q MIM execution of Prot. 4. For any $i \in [Q]$ and $k \in [\ell_{\mathsf{gad}}]$, we say that* the $\mathsf{invariant\ condition}$ holds for the k-th gadget in the i-th right session *if it holds that*

$$\Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k}] = \frac{1}{2} \pm \mathsf{negl}(\lambda),$$

where $\widetilde{a}_{i,k}$ is the value committed in the k-th ENMC *(i.e., the* ENMC *in the k-th* gadget*) in the i-th right session contained in $\mathcal{T}$, $\widetilde{b}_{i,k}$ is the value committed in the k-th* SBCom *(i.e., the* SBCom *in the k-th* gadget*) in the i-th right session contained in $\mathcal{T}$, and the probability is taken over the random procedure generating $\mathcal{T}$.*

If for all $i \in [Q]$ and $k \in [\ell_{\mathsf{gad}}]$ the $\mathsf{invariant\ condition}$ *holds for the k-th* gadget *in the i-th right session, then we simply say* the $\mathsf{invariant\ condition}$ holds.

Remark 7. W.l.o.g., we assume that if $\mathcal{A}$ decides to abort a session before it completes, the honest party simply sends the special symbol "$\perp$" as dummy messages until that session completes. Note that if $\mathcal{A}$ aborts, then the values $\widetilde{a}_{i,k}$ and $\widetilde{b}_{i,k}$ (after the abortion) are not defined. In this case, we ask the hybrid to sample $\widetilde{a}_{i,k}$ and $\widetilde{b}_{i,k}$ *uniformly at random* (so that they are defined). This will not affect the security and is only for the convenience of our presentation.

Hybrids. In the following, we define the hybrids. Along the way, we will prove two properties:

1. Each pair of adjacent hybrids are computationally indistinguishable. Looking ahead, this will allow us to argue that the simulator $\mathcal{S}$ generates $\mathcal{A}$'s view that is computationally indistinguishable to that in the real Q-Q MIM execution (i.e., Property 1 in Def. 11).

2. The $\mathsf{invariant\ condition}$ holds in all hybrids. Indeed, we only care about the $\mathsf{invariant\ condition}$ in the last hybrid because (looking ahead) it will allow us to argue that $\mathcal{A}$ cannot make V_i accept a false $\widetilde{x}_i$ *even if the left sessions are simulated by* $\mathcal{S}$ (i.e., Property 2 in Def. 11). However, due to our hybrid design, the $\mathsf{invariant\ condition}$ in a later hybrid will depend on that in earlier ones. This is why we need to establish the $\mathsf{invariant\ condition}$ in all hybrids.

In the following, we formally define these hybrids, and prove the above two properties.

Hybrid REAL**:** This is the real Q-Q MIM execution of Prot. 4. The output of this hybrid $\mathsf{OUT}(\text{REAL})$ is defined to be the internal state of $\mathcal{A}$ together with the transcript of the whole execution (i.e., all the messages exchanged on both sides).

Lemma 8. *If* SBCom *is computationally hiding and* ENMC *is extractable (as per Def. 7), then the* $\mathsf{invariant\ condition}$ *(as per Def. 15) holds in hybrid* REAL*.*

Proof. The idea for this proof is straightforward—If $\mathcal{A}$ manages to break the $\mathsf{invariant\ condition}$ for the k-th gadget in the i-th right session, then we can extract the $\widetilde{a}_{i,k}$ (due to the extractability of ENMC) and break the computationally hiding property of SBCom using $\widetilde{a}_{i,k}$ as a reasonable guess for the value $\widetilde{b}_{i,k}$ committed in the corresponding SBCom. This is exactly the same argument that we used in the proof of soundness in Sec. 4.2. We omit the details. □

Figure 1: Registers

We explain the meaning of the registers defined in hybrid $H_{0,2}$:

- $\mathtt{Ins}$ stores the instances $\{x_1, \ldots, x_Q\}$ that the left provers $\{P_1, \ldots, P_Q\}$ are going to prove;
- $\mathtt{RW}$ stores the real witnesses $\{w_1, \ldots, w_Q\}$, i.e., $w_i \in \mathcal{R}_{\mathcal{L}}(x_i)$ for all $i \in [Q]$;
- $\mathtt{TW}$ will be used to store the $\mathsf{trapdoor}$ witnesses for each left sessions; Sometimes, we will view $|\mathbf{0}\rangle_{\mathtt{TW}}$ as $\otimes_{i=1}^{Q} |\mathbf{0}\rangle_{\mathtt{TW}_i}$, where $\mathtt{TW}_i$ will be used to store the $\mathsf{trapdoor}$ witness for the i-th left session.
- $\mathtt{RT}_i$ (for $i \in [Q]$) stores a single bit indicating if the hybrid should use the real witness (when it is $|0\rangle_{\mathtt{RT}_i}$) or the $\mathsf{trapdoor}$ witness (when it is $|1\rangle_{\mathtt{RT}_i}$) in the i-th left session. (See Step 2b of $H_{j,0}$ and Step 2 of $\{H_{j,1}\}_{j\in[L]}$for details.)
- $\mathtt{AIns}$ stores the instances $\{\widetilde{x}_1, \ldots, \widetilde{x}_Q\}$ that $\mathcal{A}$ tries to prove in the Q right sessions;
- $\mathtt{Adv}$ stores $\mathcal{A}$'s internal state;
- $\mathtt{PR}$ and $\mathtt{VR}$ will be used to store the random coins used by the left provers and right verifiers respectively; Sometimes, we will view $|+\rangle_{\mathtt{PR}}$ as $\otimes_{i=1}^{Q} |+\rangle_{\mathtt{PR}_i}$ where $\mathtt{PR}_i$ stores the randomness for the i-th left prover P_i.
- $\mathtt{Tran}$ will be used to store the transcript of the MIM execution, i.e., all the messages exchanged on both sides;
- For any $j \in [L]$, $\mathtt{W}_j$ is a single-qubit register that will be used for Watrous rewinding purpose (see Step 2c of hybrid $H_{0,2}$ and Step 2 of $\{H_{j,2}\}_{j\in[L]}$ for details);
- $\mathtt{Anc}$ stores the ancilla qubits that will be used by the hybrids for purposes like purification and dilation etc.

We emphasize that all of these registers need to be initialized to proper length. To simplify the notation, we do not specify the length of them. But it is easy to see that their length is bounded by a fixed polynomial of the security parameter, and this polynomial can be determined in advance.

Hybrid $H_{0,2}$: This hybrid is essentially identical to REAL but will be described in a different way. We define this hybrid only to set up some notations and registers that will be useful in later hybrids. Formally, $H_{0,2}$ proceeds as follows:

1. **(Initialization–"Block B_0".)** On input $\{x_i\}_{i\in[Q]}$, $\{w_i\}_{i\in[Q]}$, $\{\widetilde{x}_i\}_{i\in[Q]}$[27], and $\{|\phi_i\rangle\}_{i\in[Q]}$, initialize the following state $|\psi_0\rangle$:
$$|\psi_0\rangle := |\{x_i\}_{i=1}^Q\rangle_{\mathtt{Ins}}|\{w_i\}_{i=1}^Q\rangle_{\mathtt{RW}}\,|\mathbf{0}\rangle_{\mathtt{TW}}\,(\otimes_{i=1}^Q|0\rangle_{\mathtt{RT}_i})|\{\widetilde{x}_i\}_{i=1}^Q\rangle_{\mathtt{AIns}}|\{\phi_i\}_{i=1}^Q\rangle_{\mathtt{Adv}}\,|+\rangle_{\mathtt{PR}}\,|+\rangle_{\mathtt{VR}}\,|\mathbf{0}\rangle_{\mathtt{Tran}}\,(\otimes_{j=1}^L|0\rangle_{\mathtt{W}_j})\,|\mathbf{0}\rangle_{\mathtt{Anc}}\,. \tag{14}$$
The meaning of each register is explained in Fig. 1. For notation consistency, we will refer to this step as block B_0 in later hybrids, and refer to $|\psi_0\rangle$ as the state at the end of block B_0.
2. **(Coherently Executing the Protocol.)** Starting from $|\psi_0\rangle$, finish the remaining blocks (i.e., $\{B_j\}_{j=1}^L$) in exactly the same manner as REAL but *coherently*; Meanwhile, we also perform extra work to setup the values in registers $\otimes_{j=1}^L\mathtt{W}_j$. Details are provided below.
Formally, we define a sequence of unitary operators $\{U_j\}_{j=1}^L$ as follows: For any $j\in[L]$, U_j is the dilated (or purified) procedure of the following classical[28] algorithm: Generate (and receive) messages in block B_j in the same manner as hybrid REAL with the following extra work:
   (a) **Storing Transcript in `Tran`:** When a new message is generated, put it into the next unused region in register `Tran`;[29]
   (b) **Handling WI Messages:** For any $i\in[Q]$, when a left P_i needs to generate a Stage 3 WI message, it checks the $\mathtt{RT}_i$ register and proceeds as follows:
      – If the value in $\mathtt{RT}_i$ is 0, it generates this WI message using the real witness w_i stored in `RW`. Note that this case is identical to hybrid REAL.
      – Otherwise (i.e., when it sees $|1\rangle_{\mathtt{RT}_i}$), it generates this WI message using the trapdoor witness stored in $\mathtt{TW}_i$.
      <u>Comment:</u> We remark that the case that $\mathtt{RT}_i$ contains 1 will not happen in the current hybrid, as this hybrid never modifies the $\mathtt{RT}_i$ register since its initialization. However, in later hybrids, it is possible that this case happens but $\mathtt{TW}_i$ does not contain a valid trapdoor witness for $\{(\beta_V,\mathsf{com}_{i,k},b_{i,k})\}_{k=1}^{\ell_{\mathsf{gad}}}\in\mathcal{L}_{\mathsf{match}}^{\ell_{\mathsf{gad}},\mathsf{Th}}$ (recall that $\mathcal{L}_{\mathsf{match}}^{\ell_{\mathsf{gad}},\mathsf{Th}}$ is defined in Language (6)). Looking ahead, we will prove this will not happen except for negligible probability (see Step 2 of hybrid $H_{j,1}$ and Claim 5 for more information).
   (c) **Setting Up $\mathtt{W}_j$:** At the end of block B_j, modify register $\mathtt{W}_j$ as $|0\rangle_{\mathtt{W}_j}\to|c_j\rangle_{\mathtt{W}_j}$ where c_j is defined as follows:
      – If block B_j does *not* contain any fully-nested left gadget, sample $c_j\xleftarrow{\$}\{0,1\}$ uniformly at random;
      – Otherwise (i.e., B_j contains at least one fully-nested left gadget), sample a random left gadget fully-nested in B_j, say it is the k-th gadget of the i-th left session. Then, set $c_j := \begin{cases}0 & a_{i,k}=b_{i,k}\\ 1 & a_{i,k}\neq b_{i,k}\end{cases}$, where recall that $a_{i,k}$ is the value committed in the Step 2b ENMC of the k-th gadget in the i-th left session and $b_{i,k}$ is the value committed in the Step 2a SBCom of the k-th gadget in the i-th left session (note that $b_{i,k}$ is decommitted in Step 2c).
With the $\{U_j\}_{j=1}^L$ defined above, $H_{0,2}$ in this step computes $|\psi_L\rangle=U_LU_{L-1}\cdots U_1|\psi_0\rangle$. We refer to this step as "finishing blocks $\{B_1,\ldots,B_L\}$ coherently", and refer to $|\psi_L\rangle$ as the state at the end of block B_L.
3. **(Output.)** Perform a measurement (in the computational basis) on all the registers in $|\psi_L\rangle$ except for the `Adv` register. Output the contents in registers `Adv` and `Tran`.

This finishes the description of $H_{0,2}$.

[27] Recall that Def. 11 says the $\widetilde{x}_i$'s can be adaptively chosen by the MIM $\mathcal{A}$ during the execution, but our notation here assumes that they are known in advance. One can think that the `AIns` register is initialized to 0's and the $\widetilde{x}_i$ will be put into it once $\mathcal{A}$ chooses the $\widetilde{x}_i$. This will not affect any claims in this paper.
[28] By "classical", we mean that the honest $\{P_i\}_{i\in[Q]}$ on the left, the honest $\{V_i\}_{i\in[Q]}$ on the right, and all the exchanged messages are classical. Of course, $\mathcal{A}$ could perform local quantum computation because she is a QPT adversary.
[29] Note that at the end of the execution, `Tran` will contain all the messages exchanged in the MIM execution (i.e., the full transcript).

Lemma 9. *It holds that* $\mathsf{OUT}(\mathrm{REAL}) \overset{i.d.}{=\!=} \mathsf{OUT}(H_{0,2})$. *Moreover, if the* invariant condition *(as per Def. 15) holds in* REAL*, it must hold in* $H_{0,2}$ *as well.*

Proof. The "$\mathsf{OUT}(\mathrm{REAL}) \overset{\text{i.d.}}{=\!=} \mathsf{OUT}(H_{0,2})$" part follows directly from the description of $H_{0,2}$ and the deferred measurement principle. Note that in Step 2, $H_{0,2}$ performs some extra work. But it is straightforward to see that it will not affect $\mathsf{OUT}(H_{0,2})$ at all—Step 2a only stores the transcript; Step 2b will always handle the WI messages in exactly the same manner as REAL because $\mathtt{RT}_i$ ($\forall i \in [Q]$) is initialized to 0 and remains 0 through $H_{0,2}$; Step 2c modifies the values in registers $\otimes_{j=1}^{L} \mathtt{W}_j$, but this information has no impact on other registers at all.

The "Moreover" part follows from $\mathsf{OUT}(\mathrm{REAL}) \overset{\text{i.d.}}{=\!=} \mathsf{OUT}(H_{0,2})$ and the fact that the invariant condition is fully determined by the output of the hybrid. □

Hybrid $H_{j,1}$ ($\forall j \in [L]$): This hybrid is identical to $H_{j-1,2}$, except for its behavior in block B_j. Intuitively, $H_{j,1}$ uses the same U_j as defined in $H_{0,2}$ but with the following difference: If the Stage 3 WI of some left session *starts* in block B_j, $H_{j,1}$ will use the trapdoor witness to finish this WI. Details are provided blow.

Formally, $H_{j,1}$ proceeds as follows:

1. **(Until the end of B_{j-1}.)** Proceed in the same manner as $H_{j-1,2}$ until the end of block B_{j-1}. Let $|\psi_{j-1}\rangle$ denote the state over all registers at the end of block B_{j-1}.

   Remark 8. If $j = 1$, then $|\psi_{j-1}\rangle$ is simply the $|\psi_0\rangle$ defined in Expression (14). For $j \geq 2$, this $|\psi_{j-1}\rangle$ is of the form $|\psi_{j-1}\rangle = W_{j-1}W_j \cdots W_1 |\psi_0\rangle$, where $\{W_k\}_{k=1}^{j-1}$ are unitary operators (that will be) defined in Step 2 of $\{H_{k,2}\}_{k=2}^{j-1}$.

2. **(U'_j for Block B_j.)** Compute $|\psi_j\rangle = U'_j |\psi_{j-1}\rangle$, where U'_j is defined to be the dilation of the following classical algorithm: Proceed in the same manner as the de-coherent algorithm in Step 2 of $H_{0,2}$ (i.e., the de-coherent version of U_j), but perform the following extra work:

   – If a left P_i (for some $i \in [Q]$) is required to send the *first message*[30] of the Stage 3 WI of the i-th left session, it first pauses the execution and checks if the trapdoor witness for this session is available. Note that this can be determined efficiently by check P_i's randomness stored in register $\mathtt{PR}_i$ and the transcript register $\mathtt{Tran}$. Then, it proceeds as follows:

     • If the trapdoor witness for this left session is not available, $H_{j,1}$ halts immediately and outputs a special symbol $\mathsf{Abort}_{\mathrm{NT}}$. (The "NT" stands for "no trapdoor witness").

     • Otherwise, $H_{j,1}$ records the trapdoor witness in register $\mathtt{TW}_i$, and modifies the $\mathtt{RT}_i$ register as $|0\rangle_{\mathtt{RT}_i} \rightarrow |1\rangle_{\mathtt{RT}_i}$. Then, it continues in the same manner as the de-coherent version of U_j to finish this block B_j.

     Remark 9. We make two remarks regarding this step: (i) When (right before) this Stage 3 WI starts in this block B_j, the value in $\mathtt{RT}_i$ must be 0, because this value has not been modified since its initialization; (ii) If the hybrid sets $|0\rangle_{\mathtt{RT}_i} \rightarrow |1\rangle_{\mathtt{RT}_i}$, then all the following steps (including the the remaining part of block B_j and all future blocks $\{B_{j+1}, \ldots, B_L\}$) will start to use the trapdoor witness for this i-th left session. This is by the definition of the U_j's defined in Step 2 of $H_{0,2}$.

3. **(Remaining Blocks as in $H_{j-1,2}$.)** Starting from $|\psi_j\rangle$, $H_{j,1}$ behaves identically as in $H_{j-1,2}$ and outputs whatever the latter outputs.

   Remark 10. In other wrods, this step simply computes $|\psi_L\rangle = U_L U_{L-1} \cdots U_{j+1} |\psi_j\rangle$ (where $\{U_{j+1}, \ldots, U_L\}$ are defined in Step 2 of $H_{0,2}$), performs a measurement (in the computational basis) on all the registers in $|\psi_L\rangle$ except for the $\mathtt{Adv}$ register, and outputs the contents in registers $\mathtt{Adv}$ and $\mathtt{Tran}$. (As we will define shortly, $H_{j-1,2}$ does not differ from $H_{0,2}$ in blocks $\{B_{j+1}, \ldots B_L\}$.)

This finishes the description of $H_{j,1}$.

Hybrid $H_{j,2}$ ($\forall j \in [L]$): This hybrid is identical to $H_{j,1}$, except for its behavior to handle block B_j messages. Intuitively, recall that $H_{j,1}$ executes block B_j using U'_j; In contrast, the current hybrid $H_{j,2}$ will use Watrous rewinding to perform block B_j. Formally, $H_{j,2}$ proceeds as follows:

[30] Notice that the Stage 3 WI has more than one round. It is possible that it starts in block B_j but its messages are scattered over several future blocks $B_{j'}$ where $j' > j$.

1. **(Until the end of B_{j-1}.)** Proceed in the same manner as $H_{j,1}$ until the end of block B_{j-1}. That is, compute $|\psi_{j-1}\rangle = W_{j-1}W_j \cdots W_1 |\psi_0\rangle$ (see also Rmk. 8).

2. **(Watrous Rewinding for B_j.)** Recall that unitary U'_j denotes $H_{j,1}$'s behavior for block B_j (see Step 2 of $H_{j,1}$). The current hybrid $H_{j,2}$ invokes Lem. 6 to compute $W_j = \mathsf{Amplifier}(U'_j)$ with $\mathtt{W}_j$ playing the role of the Watrous control register (recall that register $\mathtt{W}_j$ is initialized in Step 1 of $H_{0,2}$ and modified in Step 2c of $H_{j-1,2}$). Next, $H_{j,2}$ computes $|\psi_j\rangle = W_j |\psi_{j-1}\rangle$ (this is what we mean by "finishing block B_j by Watrous rewinding"). We will refer to $|\psi_j\rangle$ as the state at the end of block B_j.

3. **(Remaining Blocks as in $H_{j,1}$.)** Starting from $|\psi_j\rangle$, $H_{j,2}$ proceeds identically as $H_{j,1}$ and outputs whatever the latter outputs. That is, compute $|\psi_L\rangle = U_L U_{L-1} \cdots U_{j+1} |\psi_j\rangle$, perform a measurement (in the computational basis) on all the registers in $|\psi_L\rangle$ except for the $\mathtt{Adv}$ register, and output the contents in registers $\mathtt{Adv}$ and $\mathtt{Tran}$.

This finishes the description of $H_{j,2}$.

The following Lem. 10 and 11 establish the indistinguishability and the invariant condition in the above hybrids. The proof of these two claims will be provided in Sec. 4.4 and Sec. 4.5 respectively. For now, we make use of them to finish the current proof of Q-concurrent simulation soundness.

Lemma 10. *If* WI *is witness indistinguishable, then for any $j \in [L]$, it holds that* $\mathsf{OUT}(H_{j-1,2}) \overset{c}{\approx} \mathsf{OUT}(H_{j,1})$. *Moreover, if* SBCom *is computationally hiding,* ENMC *is both extractable (as per Def. 7) and first-message binding (as per Rmk. 4), and that the* invariant condition *holds in $H_{j-1,2}$ ($\forall j \in [L]$), then the* invariant condition *hold in $H_{j,1}$ ($\forall j \in [L]$) as well.*

Lemma 11. *If* ENMC *is computationally hiding and* WI *is witness indistinguishable, then for any $j \in [L]$, it holds that* $\mathsf{OUT}(H_{j,1}) \overset{c}{\approx} \mathsf{OUT}(H_{j,2})$. *Moreover, if we additionally assume that* ENMC *is both non-malleable and first-message binding (as per Rmk. 4), and the* invariant condition *holds in $H_{j,1}$ ($\forall j \in [L]$), then the* invariant condition *hold in $H_{j,2}$ ($\forall j \in [L]$) as well.*

Finishing the Proof of Q-Concurrent Simulation Soundness. To finish the proof of Q-concurrent simulation soundness, we need to construct a simulator $\mathcal{S}$ as required by Def. 11. Our $\mathcal{S}$ works as follows:

1. Initialize the following state $|\psi'_0\rangle$:

$$|\psi'_0\rangle := |\{x_i\}_{i=1}^Q\rangle_{\mathtt{Ins}} |\mathbf{0}\rangle_{\mathtt{RW}} |\mathbf{0}\rangle_{\mathtt{TW}} (\otimes_{i=1}^Q |0\rangle_{\mathtt{RT}_i}) |\{\widetilde{x}_i\}_{i=1}^Q\rangle_{\mathtt{AIns}} |\{\phi_i\}_{i=1}^Q\rangle_{\mathtt{Adv}} |+\rangle_{\mathtt{PR}} |+\rangle_{\mathtt{VR}} |\mathbf{0}\rangle_{\mathtt{Tran}} (\otimes_{j=1}^L |0\rangle_{\mathtt{W}_j}) |\mathbf{0}\rangle_{\mathtt{Anc}}. \tag{15}$$

2. For the remaining steps, proceed identically as $H_{L,2}$ and output whatever $H_{L,2}$ outputs.

Compare $|\psi'_0\rangle$ with the $|\psi_0\rangle$ defined Expression (14). The only difference is that the $\mathtt{RW}$ register (used to store the real witnesses for the Q left sessions) is initialized to 0's (i.e., no real witness is provided). It is straightforward that the output of $\mathcal{S}$ is identical to $\mathsf{OUT}(H_{L,2})$, because $H_{L,2}$ never uses the real witness (recall that it always used the trapdoor witness when the left Stage 3 WI starts due to the U'_j defined in Step 2 of $H_{j,1}$). In the following, we show that $\mathcal{S}$ satisfies the requirements in Def. 11.

<u>Indistinguishable Simulation.</u> It follows from Lem. 9 to 11 that $\mathsf{OUT}(\text{REAL}) \overset{c}{\approx} \mathsf{OUT}(H_{L,2})$. As discussed in the previous paragraph, $\mathsf{OUT}(H_{L,2})$ is identical to the output of $\mathcal{S}$. Therefore, it follows that the output $\mathcal{S}$ is identical to the view of $\mathcal{A}$ in the real MIM execution.

<u>Simulation Soundness.</u> Since the outputs of $\mathcal{S}$ and $H_{L,2}$ are identical, it suffices to show that (except for negligible probability over λ) $H_{L,2}$ cannot output a transcript where in the i-th (for some $i \in [Q]$) right session, the verifier V_i accepts a false statement $\widetilde{x}_i \notin \mathcal{L}$. To show that, we will use the invariant condition in $H_{L,2}$ (note that Lem. 8 to 11 imply that the invariant condition holds in $H_{L,2}$).

The intuition behind this proof is straightforward—The invariant condition implies that when the Stage 3 WI of the i-th right session starts, the trapdoor witness for this session is *not* available to the MIM adversary $\mathcal{A}$.[31] Therefore, if $\widetilde{x}_i$ is a false statement and $\mathcal{A}$ manages to convince V_i in the Stage 3 WI of this session,

[31] This can be proven using a martingale-based argument as we did in the proof of soundness in Sec. 4.2. Thus, we omit the details

then we can build from $H_{L,2}(\mathcal{A})$ a new adversary P^*_{sd} breaking the soundness of the WI—P^*_{sd} emulates the execution of $H_{L,2}$ internally where $\mathcal{A}$ is the MIM adversary, and forwards the WI messages of the i-th right session to the external verifier for the soundness game. If $\mathcal{A}$ convinces V_i on some $\widetilde{x}_i \notin \mathcal{L}$ with advantage $\delta(\lambda)$ in $H_{L,2}$, then P^*_{sd} convinces the external verifier on the same $\widetilde{x}_i \notin \mathcal{L}$ with the same advantage $\delta(\lambda)$.

But we emphasize that there is a caveat in the above argument. To build the reduction to the right Stage 3 WI, we need to define a malicious P^*_{sd} running *in straight-line*. However, in hybrid $H_{L,2}$, all the blocks $\{B_j\}_{j\in[L]}$ are executed using Watrous rewinding. Thus, the P^*_{sd} as defined above will also rewind the external verifier's messages, which is not allowed in the reduction to soundness.

To deal with this issue, we ask P^*_{sd} to guess at random in which blocks the messages of the concerned WI will appear. Since the WI has $\mathit{\Gamma}_{\text{WI}}$ rounds, P^*_{sd} will guess correctly with probability at least $(\frac{1}{L})^{\mathit{\Gamma}_{\text{WI}}}$. Assume that the WI messages are scattered in blocks $\{B_{j_1}, \ldots, B_{j_m}\}$ (for some $m \le \mathit{\Gamma}_{\text{WI}}$) and P^*_{sd} guesses these blocks correctly, we can define another hybrid $H'_{L,2}(\mathcal{A})$ which is identical to $H_{L,2}(\mathcal{A})$ except that blocks $\{B_{j_1}, \ldots, B_{j_m}\}$ are executed in straight-line (i.e., no Watrous rewindings for them any more). It is easy to show that $H'_{L,2}(\mathcal{A})$, *when conditioned on registers* $\mathtt{W}_{j_1}, \ldots, \mathtt{W}_{j_m}$ *all measuring to 0*, is equivalent to $H_{L,2}(\mathcal{A})$ (a similar argument has been used in Claim 6). Then, we can perform the above reduction with P^*_{sd} emulating $H'_{L,2}(\mathcal{A})$ internally. Notice that $H'_{L,2}(\mathcal{A})$ is equivalent to $H_{L,2}(\mathcal{A})$ with probability $(\frac{1}{2})^m$ (this is exactly the probability of registers $\mathtt{W}_{j_1}, \ldots, \mathtt{W}_{j_m}$ all measuring to 0). Thus, if $\mathcal{A}$ has advantage $\delta(\lambda)$ in $H_{L,2}(\mathcal{A})$, P^*_{sd} (first guessing blocks and then emulating $H'_{L,2}(\mathcal{A})$) will break the soundness with probability at least $\left(\frac{1}{L}\right)^{\mathit{\Gamma}_{\text{WI}}} \cdot \left(\frac{1}{2}\right)^m \cdot \delta(\lambda)$. Finally, recall from our choice of parameters that L is a polynomial of λ, and that both $\mathit{\Gamma}_{\text{WI}}$ and m ($\le \mathit{\Gamma}_{\text{WI}}$) are constants. So, the term $\left(\frac{1}{L}\right)^{\mathit{\Gamma}_{\text{WI}}} \cdot \left(\frac{1}{2}\right)^m$ is an inverse polynomial of λ. Thus, if $\delta(\lambda)$ is non-negligible, then P^*_{sd} also has a non-negligible advantage in breaking the soundness of WI.

This finishes the proof of Thm. 3.

4.4 Proving Lem. 10

4.4.1 Proving Indistinguishability

We first explain the intuition behind this proof. Notice that hybrids $H_{j-1,2}$ and $H_{j,1}$ are identical until the end of block B_{j-1}. They start to differ only if there exists some $i \in [Q]$ such that the Stage 3 WI of the i-th left session *starts* in block B_j. If that happens, $H_{j-1,2}$ proceeds using the real witness w_i, but $H_{j,1}$ proceeds using the $\mathsf{trapdoor}$ witness for this WI. Also note that starting from block B_j, the execution in both hybrids are straight-line (albeit coherently). Thus, the indistinguishability follows from the WI property of this Stage 3 WI (of the i-th left session). To make this argument formal, we need to:

1. Make the coherent execution in blocks $\{B_j, \ldots, B_L\}$ de-coherent (in the sense of Footnote 28). This is necessary if we want to build a reduction to the WI property.
2. Prove that if the Stage 3 WI of some left session really starts in block B_j, the $\mathsf{trapdoor}$ witness must be available for that session (i.e., $H_{j,1}$ does not output $\mathsf{Abort}_{\text{NT}}$ in Step 2). Otherwise, we will not be able to prepare the second witness for the reduction to the WI property.

In the following, we handle these two steps formally.

For Step 1. We define two intermediate hybrids that can be understood as $H_{j-1,2}$ and $H_{j,1}$ executed *de-coherently* for blocks $B_j, \ldots, B_L$:

- **Hybrid $H'_{j-1,2}$:** Proceed identically as $H_{j-1,2}$ until the end of block B_{j-1}. That is, compute $|\psi_{j-1}\rangle = W_{j-1}W_{j-2}\cdots W_1 |\psi_0\rangle$. Then, measure all the registers in $|\psi_{j-1}\rangle$ except for register $\mathtt{Adv}$. Next, proceed using the classical (see Footnote 28) algorithm described in Step 2 of $H_{0,2}$ (i.e., the de-coherent version of U_j's) to finish the remaining blocks $\{B_j, \ldots, B_L\}$. Finally, output $\mathcal{A}$'s internal state and the transcript of this execution. Let $\mathsf{OUT}(H'_{j-1,2})$ denote its output.
- **Hybrid $H'_{j,1}$:** Proceed identically as $H_{j,1}$ until the end of block B_{j-1}. That is, compute $|\psi_{j-1}\rangle = W_{j-1}W_{j-2}\cdots W_1 |\psi_0\rangle$. Then, measure all the registers in $|\psi_{j-1}\rangle$ except for register $\mathtt{Adv}$. Next, proceed using the classical (see Footnote 28) algorithm described in Step 2 of $H_{j,1}$ (i.e., the de-coherent version of U'_j) to finish block B_j. Then, proceed using the classical (see Footnote 28) algorithm described in Step 2 of $H_{0,2}$ to finish the remaining blocks $\{B_{j+1}, \ldots, B_L\}$. Finally, output $\mathcal{A}$'s internal state and the transcript of this execution. Let $\mathsf{OUT}(H'_{j,1})$ denote its output.

It follows directly from the deferred measurement principle that $\mathsf{OUT}(H'_{j-1,2}) \overset{\text{i.d.}}{=\!=} \mathsf{OUT}(H_{j-1,2})$ and $\mathsf{OUT}(H'_{j,1}) \overset{\text{i.d.}}{=\!=} \mathsf{OUT}(H_{j,1})$. Thus, we only need to show that $\mathsf{OUT}(H'_{j-1,2}) \overset{\text{c}}{\approx} \mathsf{OUT}(H'_{j,1})$.

For Step 2. Let E_{NT} denote the event that hybrid $H'_{j,1}$ outputs $\mathsf{Abort}_{\text{NT}}$ in Step 2. As mentioned above, we need to prove the following claim:

Claim 5. *It holds that* $\Pr_{H'_{j,1}}[E_{\text{NT}}] \le \mathsf{negl}(\lambda)$.

Proof. This claim follows from a similar proof template from [ACL21], where the authors also need to prove that the trapdoor witness for a session is available, i.e., there are enough "slots" (our gadgets in their term) are matching, when the hybrid needs to perform the WI of that session. Since our derivation is slightly different from [ACL21], we still provide this proof (in Appx. B). □

With Claim 5, the indistinguishability $\mathsf{OUT}(H'_{j-1,1}) \overset{\text{c}}{\approx} \mathsf{OUT}(H'_{j,2})$ can be reduced to the WI property of the Stage 3 WI as explained at the beginning of Sec. 4.4.1. Since this reduction is standard, we omit the details.

4.4.2 Proving Invariant Condition

We keep using the hybrids $H'_{j-1,2}$ and $H'_{j,1}$ defined in Sec. 4.4.1. Since $\mathsf{OUT}(H_{j,1}) \overset{\text{i.d.}}{=\!=} \mathsf{OUT}(H'_{j,1})$, it suffices to prove that in hybrid $H'_{j,1}$, for any $i \in [Q]$ and $k \in [\ell_{\mathsf{gad}}]$, the invariant condition holds for k-th gadget of the i-th right session. For any $i \in [Q]$ and $k \in [\ell_{\mathsf{gad}}]$, we will use the final message of block B_{j-1} as a pivot to divide all possible schedules into the following four cases (illustrated in Fig. 2) and show the invariant condition for each of them separately:

1. The decommitment to $\widetilde{b}_{i,k}$ ends before (or in parallel with) the final message of B_{j-1} (illustrated in Fig. 2a);

2. The SBCom to $\widetilde{b}_{i,k}$ starts after the final message of B_{j-1} (illustrated in Fig. 2b);

3. The SBCom to $\widetilde{b}_{i,k}$ ends before (or in parallel with) the final message of B_{j-1}, but the ENMC to $\widetilde{a}_{i,k}$ starts after the final message of B_{j-1} (illustrated in Fig. 2c);

4. The ENMC to $\widetilde{a}_{i,k}$ starts before (or in parallel with) the final message of B_{j-1}, but this ENMC *but ends after* final message of B_{j-1} (illustrated in Fig. 2d).

For Case 1: This is an easy case. Note that $H'_{j-1,2}$ and $H'_{j,1}$ are identical until the end of block B_{j-1}, by when the invariant condition for the k-th gadget of the i-th right session is already determined (for Case 1 schedules). By the condition in the statement of Lem. 10, the invariant condition holds in $H'_{j-1,2}$. Thus, it must hold in in $H'_{j,1}$ as well.

For Case 2: This is also an easy case. Note that after the final message of the block B_{j-1}, the execution of $H'_{j,1}$ is in straight-line. Thus, using the same argument as in the proof of Lem. 8, it is easy to see that the invariant condition follows from the computationally hiding property of SBCom and the extractability of ENMC.

For Case 3: We want to handle this case in the same manner as for Case 2. That is, if $\mathcal{A}$ manages to violate the invariant condition, we can build a new adversary $\mathcal{A}_{\mathsf{hd}}$ breaking the computational hiding property of the SBCom. In more detail, $\mathcal{A}_{\mathsf{hd}}$ uses the external hiding challenger's SBCom in place of SBCom to $\widetilde{b}_{i,k}$, extracts $\widetilde{a}_{i,k}$ using the extractor for ENMC, and uses this extracted $\widetilde{a}_{i,k}$ as a reasonable guess for the secret bit committed by the external hiding challenger.

However, there is one difference from Case 2 that stops us from doing so—The SBCom to $\widetilde{b}_{i,k}$ happens within some block B_z with $z \le j-1$; This B_z is performed using Watrous rewinding. Thus, we cannot view the this right gadget as a straight-line execution and perform the same reduction to the hiding property of SBCom.

To resolve this issue, we define another hybrid $H''_{j,1}(\mathcal{A})$ which is identical to $H'_{j,1}(\mathcal{A})$ except that block B_z is executed in straight-line. Thus, $H''_{j,1}(\mathcal{A})$ *conditioned on register* $\mathtt{W}_z$ *measuring to 0* is equivalent to the original $H'_{j,1}(\mathcal{A})$ (a similar argument has been used in Claim 6). We can perform the above reduction in

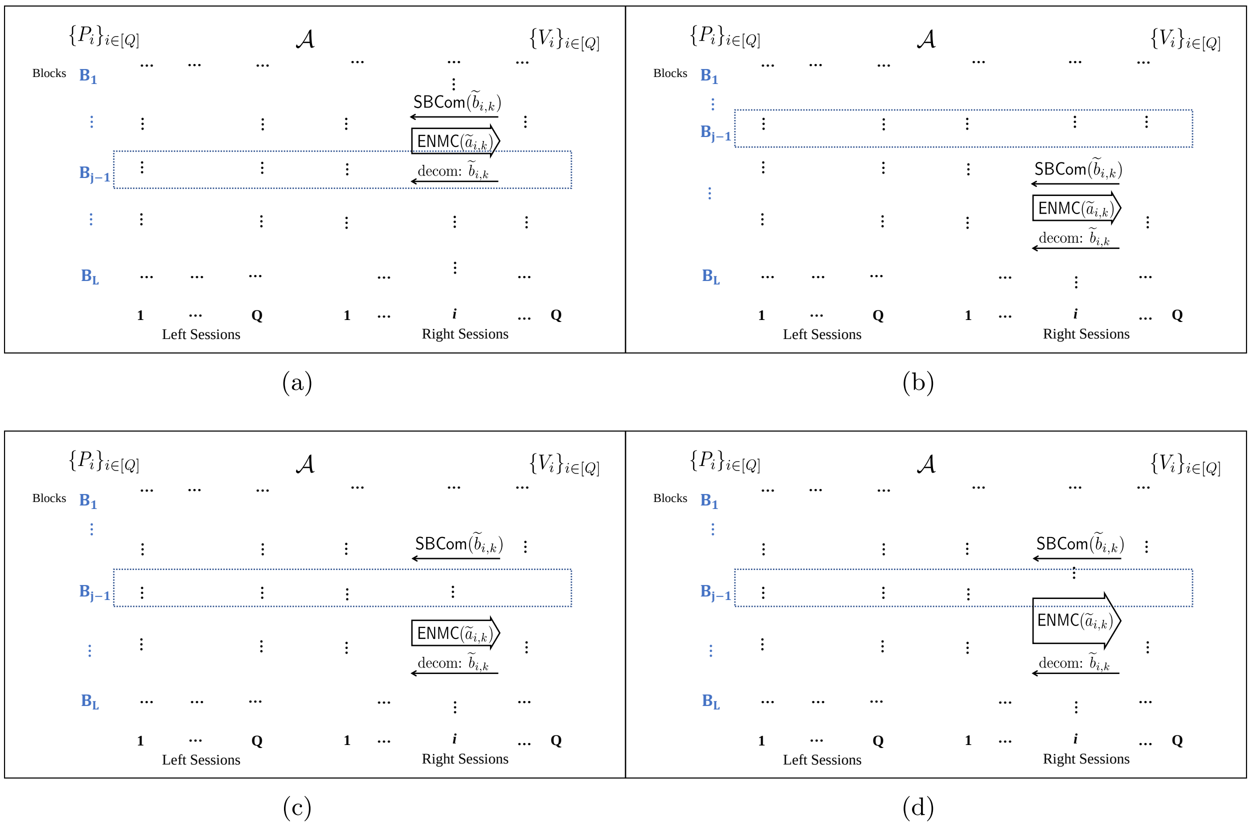


Fig. 2: Different Schedules

$H''_{j,1}(\mathcal{A})$. Notice that $H''_{j,1}(\mathcal{A})$ is equivalent to $H'_{j,1}(\mathcal{A})$ with probability 1/2 (this is exactly the probability of register $\mathtt{W}_z$ measuring to 0). Thus, if $\mathcal{A}_{\mathsf{hd}}$ has advantage $\geq \frac{1}{2} + \delta(\lambda)$ using $H'_{j,1}(\mathcal{A})$, then she has advantage $\geq \frac{1}{2} + \frac{\delta(\lambda)}{2\ell_{\mathsf{gad}}}$ using $H''_{j,1}(\mathcal{A})$, which still suffices to break the hiding property of SBCom. (The term $\frac{1}{\ell_{\mathsf{gad}}}$ is to account for the fact that $\mathcal{A}_{\mathsf{hd}}$ needs to guess correctly in which block B_z the SBCom to $\widetilde{b}_{i,k}$ will appear.)

For Case 4: The invariant condition in this case follows from the first-message binding property of the ENMC (to $\widetilde{a}_{i,k}$) and the indistinguishability $H'_{j-1,2} \overset{\text{c}}{\approx} H'_{j,1}$. To see it, recall that $H'_{j-1,2}$ and $H'_{j,1}$ are identical until the end of block B_{j-1}. Thus, the value statistically-bound in the first message of ENMC (see Rmk. 4) remains unchanged when we switch from $H'_{j-1,2}$ to $H'_{j,1}$. Next, recall that the value committed in ENMC is defined by the value statistically bound in the first message of ENMC *together with the receiver's final decision*. That is, if we denote the value statistically bound in the first message as $\widetilde{a}_{i,k}$, then the committed value in ENMC is $\mathsf{Val}_d(\widetilde{a}_{i,k}) := \begin{cases} \widetilde{a}_{i,k} & d = 1 \\ \bot & d = 0 \end{cases}$, where d denotes if the receiver accepts ($d = 1$) or rejects ($d = 0$) this ENMC. Also note that this bit d can be derived efficiently from the transcript of this ENMC execution. By the conditions in the statement of Lem. 10, the invariant condition holds in $H'_{j-1,2}$. Thus, it must hold in $H'_{j,1}$ as well; Otherwise, it breaks the computational indistinguishability between these two hybrids (by checking the value d).

4.5 Proving Lem. 11

4.5.1 Proving Indistinguishability

Hybrids $H_{j,1}$ and $H_{j,2}$ differ only at block B_j, where $H_{j,2}$ performs Watrous rewinding over $H_{j,1}$'s strategy. Thus, the indistinguishability $\mathsf{OUT}(H_{j,1}) \overset{\text{c}}{\approx} \mathsf{OUT}(H_{j,2})$ follows from the Watrous' rewinding lemma (and the computational hiding property of ENMC via a standard argument (almost identical to that in [ACL21, Section 4.2.1]). For the sake of completeness, we provide the formal proof in Appx. C. In the following, we establish the invariant condition in $H_{j,2}$.

4.5.2 Proving Invariant Condition

We define an intermediate hybrid $H'_{j,1}$ that proceeds as follows:

1. **(Until the End of B_{j-1}.)** Proceed identically as $H_{j,1}$ until the end of block B_{j-1}. That is, compute $|\psi_{j-1}\rangle = W_{j-1} \cdots W_1 |\psi_0\rangle$. Then, it measures all the registers in $|\psi_{j-1}\rangle$ except for register $\mathtt{Adv}$.
2. **(De-coherently Executing B_j.)** Proceed using the classical (see Footnote 28) algorithm described in Step 2 of $H_{0,1}$ (i.e., the de-coherent version of U'_j) to finish block B_j.
3. **(De-coherently Executing $\{B_{j+1}, \ldots, B_L\}$.)** Proceed using the classical (see Footnote 28) algorithm described in Step 2 of $H_{0,2}$ (i.e., the de-coherent version of $\{U_{j+1}, \ldots, U_L\}$) to finish the remaining blocks $\{B_{j+1}, \ldots, B_L\}$. Finally, output $\mathcal{A}$'s internal state and the transcript of this execution.

Similarly, we also define an intermediate hybrid $H'_{j,2}$ that proceeds as follows:

1. **(Until the End of B_{j-1}.)** Proceed identically as $H_{j,2}$ until the end of block B_{j-1}. That is, compute $|\psi_{j-1}\rangle = W_{j-1} \cdots W_1 |\psi_0\rangle$.
2. **(Watrous Rewinding for B_j.)** Compute $|\psi_j\rangle = W_j |\psi_{j-1}\rangle$. Then, it measures all the registers in $|\psi_j\rangle$ except for register $\mathtt{Adv}$.
3. **(De-coherently Executing $\{B_{j+1}, \ldots, B_L\}$.)** Proceed using the classical (see Footnote 28) algorithm described in Step 2 of $H_{0,2}$ (i.e., the de-coherent version of $\{U_{j+1}, \ldots, U_L\}$) to finish the remaining blocks $\{B_{j+1}, \ldots, B_L\}$. Finally, output $\mathcal{A}$'s internal state and the transcript of this execution.

It follows directly from the deferred measurement principle that $\mathsf{OUT}(H'_{j,1}) \overset{\text{i.d.}}{=\!=} \mathsf{OUT}(H_{j,1})$ and $\mathsf{OUT}(H'_{j,2}) \overset{\text{i.d.}}{=\!=} \mathsf{OUT}(H_{j,2})$. Thus, to establish the invariant condition in $H_{j,2}$, it suffices to prove the it in hybrid $H'_{j,2}$.

Toward that, we first define an event E_0 in hybrid $H'_{j,1}$:

– **Event E_0:** At the end of block B_j, the $\mathtt{W}_j$ register contains value 0.

We prove a useful claims regarding event E_0.

Claim 6. *For any QPT adversary $\mathcal{A}$, it holds that:*

$$\forall i \in [Q], \forall k \in [\ell_{\mathsf{gad}}], \quad \Pr_{H'_{j,2}(\mathcal{A})}[\widetilde{a}_{i,k} = \widetilde{b}_{i,k}] \overset{s}{\approx} \Pr_{H'_{j,1}(\mathcal{A})}[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid E_0],$$

where $H'_{j,1}(\mathcal{A})$ (resp. $H'_{j,2}(\mathcal{A})$) denotes the execution of $H'_{j,1}$ (resp. $H'_{j,2}$) with $\mathcal{A}$ acting as the MIM adversary.

Proof. Recall that $H'_{j,1}$ and $H'_{j,2}$ only differ at block B_j, where $H'_{j,1}$ uses the (de-coherent version of) U'_j but $H'_{j,2}$ use Watrous rewinding circuit W_j. Thus, Claim 6 holds as long as the following holds:

$$W_j|\psi_{j-1}\rangle \overset{s}{\approx} |\psi^0_j\rangle|0\rangle_{\mathtt{W}_j}, \tag{16}$$

where $|\psi^0_j\rangle|0\rangle_{\mathtt{W}_j}$ is the branch correpsonding to $\mathtt{W}_j$ containing 0 in the superposition of $U'_j|\psi_{j-1}\rangle$.

We remark that Eq. (16) is a direct result of the Watrous rewinding lemma, which says the effect of Watrous' rewinding is to "kills" the branch corresponding to the Watrous control register bing 1, and only retains the branch corresponding to the Watrous control register bing 0. Indeed, this Eq. (16) is formally established as Eq. (85) when we prove the indistinguishability between $H_{j,1}$ and $H_{j,2}$ using the Watrous' rewinding lemma. We refer to Appx. C for details.

This finishes the proof of Claim 6. □

It follows from Claim 6 that to show the invariant condition in $H'_{j,2}$, we only need to show it in $H'_{j,1}$ *conditioned on* E_0. Formally, we need to prove that for any QPT adversary $\mathcal{A}$, it holds that

$$\forall i \in [Q], \forall k \in [\ell_{\mathsf{gad}}], \quad \Pr_{H'_{j,1}}[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid E_0] = \frac{1}{2} \pm \mathsf{negl}(\lambda). \tag{17}$$

The remainder of this proof is devoted to establishing Eq. (17).

Fix any $i \in [Q]$ and $k \in [\ell_{\mathsf{gad}}]$, we divide all possible schedules into the following two types:

1. **Type-1**: The first message of ENMC to $\widetilde{a}_{i,k}$ starts before (or in parallel with) the last message of block B_{j-1};
2. **Type-2**: The first message of ENMC to $\widetilde{a}_{i,k}$ starts after the last message of block B_{j-1}.

First, we claim that Eq. (17) holds for **Type-1** schedules. Recall that ENMC is first-message binding (see Rmk. 4). That is, the first message of ENMC already fixed the value $\widetilde{a}_{i,k}$. Thus, whether E_0 happens or not (which depends only on the execution *after* the first message of ENMC to $\widetilde{a}_{i,k}$ for **Type-1** schedule) does not affect the probability of "$\widetilde{a}_{i,k} = \widetilde{b}_{i,k}$"[32]. Therefore, the following holds for **Type-1** schedules:

$$\Pr_{H'_{j,1}}\left[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid E_0\right] = \Pr_{H'_{j,1}}\left[\widetilde{a}_{i,k} = \widetilde{b}_{i,k}\right] = \Pr_{H_{j,1}}\left[\widetilde{a}_{i,k} = \widetilde{b}_{i,k}\right] = \frac{1}{2} \pm \mathsf{negl}(\lambda), \tag{18}$$

where the last equation follows from the $\mathsf{invariant\ condition}$ in $H_{j,1}$.

Therefore, in the following, we only need to prove Eq. (17) for **Type-2** schedules. To do that, we further divide **Type-2** schedules into two sub-types, based on if there exist fully nested left $\mathsf{gadgets}$ in block B_j:

– **Type-2-1:** these are **Type-2** schedules where there are no fully nested left $\mathsf{gadgets}$ in block B_j;

– **Type-2-2:** these are **Type-2** schedules where there exists at least one fully nested left $\mathsf{gadgets}$ in block B_j;

For **Type-2-1** schedules, recall that the the $\mathsf{Watrous\ control\ register}$ is set to be c_j sampled *uniformly at random* from $\{0, 1\}$ (because there are no fully nested left $\mathsf{gadgets}$ in B_j). Thus, whether E_0 happens or not is independent to other parts of the execution for **Type-2-1** schedules. In particular, E_0 will not affect the event "$\widetilde{a}_{i,k} = \widetilde{b}_{i,k}$" for **Type-2-1** schedules. Therefore, in this case, Eq. (17) simply follows from the $\mathsf{invariant\ condition}$ in $H_{j,1}$.

In the following, we only need to focus on **Type-2-2** schedules. In this case, we will reduce Eq. (17) to the non-malleability of ENMC. Indeed, this is the only interesting type of schedules. We show the result by the following Claim 7, whose proof is given in Sec. 4.5.3.

Claim 7. *Assume the* $\mathsf{invariant\ condition}$ *holds in* $H_{j,1}$ *and the* ENMC *is non-malleable. Then* Eq. (17) *holds for* **Type-2-2** *schedules.*

4.5.3 Proving Claim 7

First, note that for **Type-2-2** schedules, event E_0 means that a randomly-picked fully nested left gadget in B_j matches. Also, even for **Type-2-2** schedules, we know that E_0 must happen with probability $\frac{1}{2}$. This is because that the picked left gadget matches with probability $\frac{1}{2}$ (since the left honest prover always picks the value $\widetilde{a}_{i,k}$ uniformly at random in Step 2b of Prot. 4). That is, the following holds for **Type-2-2** schedules:

$$\Pr_{H'_{j,1}(\mathcal{A})}[E_0] = \Pr_{H'_{j,1}(\mathcal{A})}[\neg E_0] = \frac{1}{2}. \tag{19}$$

Moreover, for any $i \in [Q]$ and $k \in [\ell_{\mathsf{gad}}]$, the following holds for **Type-2-2** schedules (we omit $H'_{j,1}(\mathcal{A})$ from the subscript of "Pr" for succinctness)

$$\Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid E_0] \cdot \Pr[E_0] + \Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid \neg E_0] \cdot \Pr[\neg E_0] = \Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k}] = \frac{1}{2} \pm \mathsf{negl}(\lambda), \tag{20}$$

where Eq. (20) follows from the $\mathsf{invariant\ condition}$ in $H_{j,1}$.

Eq. (19) and (20) together imply that

$$\forall i \in [Q], \forall k \in [\ell_{\mathsf{gad}}],\ \ \Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid E_0] + \Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid \neg E_0] = 1 \pm \mathsf{negl}(\lambda) \tag{21}$$

[32] Actually, it is possible that $\mathcal{A}$ decides to abort the execution of ENMC to $\widetilde{a}_{i,k}$ depending on even E_0, which leads to $\widetilde{a}_{i,k} = \bot$. But this will not affect Eq. (18) since in that case, $\widetilde{a}_{i,k}$ and $\widetilde{b}_{i,k}$ will be re-defined to be independently sampled bits (see Rmk. 7), for which $\Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k}]$ is exactly $\frac{1}{2}$.

In the following, we prove Claim 7 by a reduction to the non-malleability of ENMC. Formally, we assume for contradiction that Claim 7 is false, i.e., there exist an $i \in [Q]$, a $k \in [\ell_{\mathsf{gad}}]$, and a $\delta(\lambda) = \frac{1}{\mathsf{poly}(\lambda)}$ such that the following holds for infinitely many λ

$$\big|\Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid E_0] - \frac{1}{2}\big| \geq \delta(\lambda). \tag{22}$$

For such λ's, it follows from Eq. (21) and (22) that

$$\big|\Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid E_0] - \Pr[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid \neg E_0]\big| \geq 2\delta(\lambda) - \mathsf{negl}(\lambda). \tag{23}$$

Next, we build a QPT adversary $\mathcal{A}_{\text{NM}}$ and the associated QPT distinguisher $\mathcal{D}_{\text{NM}}$ that breaks the (non-uniform) non-malleability of ENMC (recall it from Def. 8).

Recall that for **Type-2-2** schedules, the event E_0 means the following: Picking a random left gadget that is fully nested in block B_j (say, it is the v-th gadget of the u-th left session), the value $b_{u,v}$ committed by $\mathcal{A}$ in SBCom is equal to the value $a_{u,v}$ committed by the left honest prover in ENMC. With this observation, $\mathcal{A}_{\text{NM}}$ can be designed as follows:

- $\mathcal{A}_{\text{NM}}$ internally emulates the game $H'_{j,1}(\mathcal{A})$ until the moment when $\mathcal{A}$ sends SBCom to $b_{u,v}$, pauses the execution, and performs brute-force search to learn the value $b_{u,v}$. We remark that this step is not efficient. But it happens before $\mathcal{A}_{\text{NM}}$ starts participating in the non-malleability game (which starts from next step). Thus, the information in this step can be thought as a *non-uniform* advice to $\mathcal{A}_{\text{NM}}$.
- $\mathcal{A}_{\text{NM}}$ starts to participate in the non-malleability game: She sets $m_0 := b_{u,v}$ and $m_1 := 1 - b_{u,v}$, and sends (m_0, m_1) to the external non-malleability challenger Ch. Ch will flip a random coin b and performs a MIM execution of ENMC with $\mathcal{A}_{\text{NM}}$ where on the left side, Ch commits to m_b as an honest committer, and on the right side, Ch acts as an honest receiver, expecting to receive an ENMC to some $\widetilde{m}$ with $\mathcal{A}_{\text{NM}}$ acting as a (potentially malicious) committer.
- $\mathcal{A}_{\text{NM}}$ participates in this game by relaying messages between the external Ch and the internal $\mathcal{A}$. In particular, she will use Ch's messages on the left side as the ENMC to $a_{u,v}$ (i.e., $a_{u,v} := m_b$), and use $\mathcal{A}$'s ENMC to $\widetilde{a}_{i,k}$ as the messages in the right execution with Ch (i.e., $\widetilde{m} := \widetilde{a}_{i,k}$). All other messages are still emulated by $\mathcal{A}_{\text{NM}}$ internally.

The distinguisher $\mathcal{D}_{\text{NM}}$ works as follows: It first checks if $\widetilde{m}$ is equal to $\widetilde{b}_{i,k}$. Recall that $\widetilde{m}$ is the value committed by $\mathcal{A}_{\text{NM}}$ to Ch in the MIM execution of ENMC (Indeed, this is exact the ENMC to $\widetilde{a}_{i,k}$ made by the $\mathcal{A}$ internally emulated by $\mathcal{A}_{\text{NM}}$). If $\widetilde{m} = \widetilde{b}_{i,k}$, $\mathcal{D}_{\text{NM}}$ outputs 1; Otherwise, it outputs 0.

To analyze $(\mathcal{A}_{\text{NM}}, \mathcal{D}_{\text{NM}})$'s advantage, notice that if Ch commits to $m_0 = b_{u,v}$, then the internal $\mathcal{A}$'s view is identical to $H'_{j,1}(\mathcal{A})$ conditioned on $a_{u,v} = b_{u,v}$ (aka, conditioned on E_0). Also recall that $\mathcal{D}_{\text{NM}}$ by definition outputs 1 iff $\widetilde{a}_{i,k} = \widetilde{m}$ $(= \widetilde{b}_{i,k})$. In this case, it implies

$$\Pr_{\mathsf{mim}_0}\big[\mathsf{OUT}(\mathcal{D}_{\text{NM}}) = 1\big] = \Pr_{H'_{j,1}(\mathcal{A})}[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid E_0], \tag{24}$$

where mim_0 denotes the non-malleability game where Ch commits to m_0. By a symmetric argument, we can show that

$$\Pr_{\mathsf{mim}_1}\big[\mathsf{OUT}(\mathcal{D}_{\text{NM}}) = 1\big] = \Pr_{H'_{j,1}(\mathcal{A})}[\widetilde{a}_{i,k} = \widetilde{b}_{i,k} \mid \neg E_0], \tag{25}$$

where mim_1 corresponds to the case where Ch commits to $1 - b_{u,v} = m_1$ $(= a_{u,v})$ (Therefore, $a_{u,v} \neq b_{u,v}$ and thus $\neg E_0$.)

Eq. (24) and (25) together with Eq. (23) imply that $\mathcal{D}_{\text{NM}}$ has non-negligible advantage $2\delta(\lambda) - \mathsf{negl}(\lambda)$ in the non-malleability game, thus completing the proof.

Remark 11. The above argument assumed that $\mathcal{A}_{\text{NM}}$ knows which left gadget u-v is picked to determine E_0 (so that $\mathcal{A}_{\text{NM}}$ can use this left u-v gadget and the concerned right i-k gadget to finish the above reduction to non-malleability). However, for **Type-2-2** schedules, $\mathcal{A}_{\text{NM}}$ only knows there exists at least one fully nested left gadget in B_j, but she does not known *which fully nested left* gadget *is the* u-v *that determines* E_0.

This issue can be handled as follows. We simply ask $\mathcal{A}_{\text{NM}}$ to guess *randomly* in advance which left gadget will be used as the u-v left gadget, and behaves as if she guessed correctly using the above reduction. Note that there are only polynomially many left $\mathsf{gadgets}$ in total (the accurate number is $Q \cdot \ell_{\mathsf{gad}}$). Thus, $\mathcal{A}_{\text{NM}}$ will guess correctly with probability $\frac{1}{Q \cdot \ell_{\mathsf{gad}}}$.[33] Thus, this random guess will only introduce a $\frac{1}{Q \cdot \ell_{\mathsf{gad}}}$ multiplicative factor over $(\mathcal{A}_{\text{NM}}, \mathcal{D}_{\text{NM}})$'s eventual advantage. So the above reduction still go through.

This finishes the proof of Claim 7.

4.6 Extension to QMA

As discussed in the technical overview, we remark that the proof of Thm. 3 makes use of the Stage 3 WI of Prot. 4 in a "black-box" manner. That is, all the claims/lemmas above hold as long as the Stage 3 WI is constant-round, computationally WI, and computationally sound. In particular, this is true even if this Stage 3 WI involves quantum communication.

Therefore, a bounded-concurrent simulation-sound ZK argument for **QMA** can be constructed by replacing the Stage 3 WI for **NP** in Prot. 4 with a WI argument for **QMA**, where again the prover proves *either* the **QMA** statement is true *or* the $\mathsf{trapdoor}$ statement is true. (We refer to [CCLY22, Section 6.4] for definitions of **QMA**, WI arguments for **QMA**, etc.) We remark that such a constant-round WI argument for **QMA** is known from the existence of PQ-OWFs in [CCLY22]. In more detail, [CCLY22] constructed a constant-round ε-ZK argument for **QMA** using only PQ-OWFs. It is well-known that ε-ZK implies WI. This leads to the following theorem:

Theorem 8. *Assuming the existence of PQ-OWFs, bounded-concurrent simulation-sound ZK arguments for* **QMA** *exist.*

5 Bounded-Concurrent Post-Quantum Two-Party Coin-Flipping Protocols

5.1 Construction

Our construction makes use of all the building blocks for our bounded-concurrent simulation-sound ZK protocol in Prot. 4, i.e., SBCom, ENMC, and WI. Additional, we need the following extra building block:

– A two-round[34] function-hiding PQ-SFE scheme $\mathsf{SFE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ (as per Def. 9). Let $\mathit{\Gamma}_{\text{SFE}}$ denote the round complexity of SFE. (In our case, $\mathit{\Gamma}_{\text{SFE}}$ is equal to 2.)

Our bounded-concurrent post-quantum two-party coin-flipping protocol is presented in Prot. 5. (Similar as in Prot. 4, we assume w.l.o.g. that all the instances of Naor's commitment SBCom is non-interactive.) We explain the purpose of each step in the following "***Comment***" environment.

Protocol 5: Bounded-Concurrent Post-Quantum Two-Party Coin-Flipping Protocol

Let $Q(\lambda)$ be a polynomial of λ, denoting the maximum number of concurrent sessions. P_1 and P_2 take the security parameter λ and an ID $\mathsf{id} \in \{0,1\}^\lambda$ as the common input.

1. **(Com-and-Guess: P_1 is Sender.)** For $k = 1$ to $\ell_{\mathsf{gad}} := 120Q^7\lambda$:
   (a) P_2 samples $b_{1,k} \xleftarrow{\$} \{0,1\}$ and commits to it using SBCom.
   (b) P_1 samples $a_{1,k} \xleftarrow{\$} \{0,1\}$ and commits to it using ENMC. P_1 and P_2 will use $\mathsf{id}{:}1{:}k$ as the ID for this ENMC so that each ENMC is executed using a different ID.
   (c) P_2 sends $b_{1,k}$ together with the decommitment information w.r.t. the SBCom in Step 1a. P_1 continues only if this decommitment is valid.

[33] Actually, $\mathcal{A}_{\text{NM}}$ only needs to guess within the block B_j. Her probability of correct guess is lower-bounded by the inverse of the max number of left $\mathsf{gadgets}$ that B_j could possibly contain. This probability is even higher than $\frac{1}{Q \cdot \ell_{\mathsf{gad}}}$. But we choose to use $\frac{1}{Q \cdot \ell_{\mathsf{gad}}}$ as it already suffices for our purpose.

[34] Actually, any constant-round construction suffices.

Comment. *This step is identical to Stage 2 of Prot. 4. Similarly, we call each repetition a Stage 1* gadget*. The k-th Stage 1* gadget matches *if the $a_{1,k}$ committed by P_1 in Step 1b is equal to the $b_{1,k}$* validly *decommitted by P_2 in Step 1c. This step is designed to allow the simulator to learn a* $\mathsf{trapdoor}$ *witness (i.e., there are more than* $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ *Stage 1* $\mathsf{gadgets}$ *matching) when P_2 is corrupted, while ensuring that the corrupted P_2 cannot learn the* $\mathsf{trapdoor}$ *witness. We say that P_1 is the* sender *and P_2 is the* $\mathsf{receiver}$ *of this* **Com-and-Guess Stage**. *This terminology will be useful in Sec. 5.2.*

2. **(Com-and-Guess: P_2 is Sender.)** For $k = 1$ to $\ell_{\mathsf{gad}} := 120Q^7\lambda$:
   (a) P_1 samples $b_{2,k} \xleftarrow{\$} \{0,1\}$ and commits to it using SBCom.
   (b) P_2 samples $a_{2,k} \xleftarrow{\$} \{0,1\}$ and commits to it using ENMC. P_1 and P_2 will use $\mathsf{id}{:}2{:}k$ as the ID for this ENMC.
   (c) P_1 sends $b_{2,k}$ together with the decommitment information w.r.t. the SBCom in Step 2a. P_2 continues only if this decommitment is valid.

   Comment. *This step is designed to allow the simulator to learn a* $\mathsf{trapdoor}$ *witness (as the simulator for Prot. 4 does) when P_1 is corrupted, while ensuring that the corrupted P_1 cannot learn any* $\mathsf{trapdoor}$ *witness. Similar as in Stage 2 of Prot. 4, We call each repetition a Stage 2* gadget*. The k-th Stage 2* gadget matches *if the $a_{2,k}$ committed by P_2 in Step 2b is equal to the $b_{2,k}$* validly *decommitted by P_1 in Step 2c. We say that P_2 is the* sender *and P_1 is the* $\mathsf{receiver}$ *of this* **Com-and-Guess Stage**. *This terminology will be useful in Sec. 5.2.*

3. **(Commitment to P_1's Share.)** P_1 samples a random string r_1. P_1 commits to r_1 using the statistically-binding commitment SBCom.
4. **(Commitment to P_2's Share.)** P_2 samples a random string r_2. P_2 commits to r_2 using the statistically-binding commitment SBCom.
5. **(SFE: P_1 is Receiver.)** We first define a classical circuit C_1:
   – **Circuit C_1:** It has r_2 and the transcript of Stage 1 hard-wired in; It takes as input a string w. If w is a witness for the fact that there are more than $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ Stage 1 $\mathsf{gadgets}$ matching, it outputs r_2; Otherwise, it outputs a dummy symbol $\vdash$.

   P_1 computes $\mathsf{k}_1 \leftarrow \mathsf{SFE.Gen}(1^\lambda)$ and $\mathsf{ct}_1 \leftarrow \mathsf{SFE.Enc}(\mathsf{k}_1, w_1)$, where w_1 is set to an all-0 string. P_1 sends ct_1 to P_2. P_2 computes $\widehat{\mathsf{ct}}_1 \leftarrow \mathsf{SFE.Eval}(C_1, \mathsf{ct}_1)$ and sends $\widehat{\mathsf{ct}}_1$ to P_1. P_1 simply ignores the message $\widehat{\mathsf{ct}}_1$.

   Comment. *This step is designed to allow the simulator learn P_2's committed share r_2, when P_2 is corrupted. In that case, the simulator will set w_1 to the* $\mathsf{trapdoor}$ *witness it learned from Stage 1, so that she can learn r_2 by decrypting $\widehat{\mathsf{ct}}_1$. Note that an honest P_1 will always get $\vdash$ when decrypting $\widehat{\mathsf{ct}}_1$ (as w_1 is set to an all-0 string). So the message $\widehat{\mathsf{ct}}_1$ is useless for an honest P_1.*

6. **(SFE: P_2 is Receiver.)** We first define a classical circuit C_2:
   – **Circuit C_2:** It has r_1 and the transcript of Stage 2 hard-wired in; It takes as input a string w. If w is a witness for the fact that there are more than $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ Stage 2 $\mathsf{gadgets}$ matching, it outputs r_1; Otherwise, it outputs a dummy symbol $\vdash$.

   P_2 computes $\mathsf{k}_2 \leftarrow \mathsf{SFE.Gen}(1^\lambda)$ and $\mathsf{ct}_2 \leftarrow \mathsf{SFE.Enc}(\mathsf{k}_2, w_2)$, where w_2 is set to an all-0 string. P_2 sends ct_2 to P_1. P_1 computes $\widehat{\mathsf{ct}}_2 \leftarrow \mathsf{SFE.Eval}(C_2, \mathsf{ct}_2)$ and sends $\widehat{\mathsf{ct}}_2$ to P_2. P_2 simply ignores the message $\widehat{\mathsf{ct}}_2$.

   Comment. *This step is symmetric to Stage 5.*

7. **(P_1 Reveals r_1.)** P_1 sends the r_1 that is committed in Stage 3. We emphasize that P_1 only sends the value r_1, *without the associated decommitment information.*
8. **(P_2 Reveals r_2.)** P_2 sends the r_2 that is committed in Stage 4. We emphasize that P_2 only sends the value r_2, *without the associated decommitment information.*
9. **(P_1 Proves Honesty.)** P_1 and P_2 execute an instance of WI where P_1 proves that

(a) *either* it behaves honestly by following the instructions in Stages 3, 5 and 7; *or*

(b) there are more that $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ Stage 1 gadgets matching.

We remark that an honest P_1 will use the witness for Item 9a to finish this WI.

Comment. *This step is designed to protect against a corrupted P_2. In that case, the simulator will use the* trapdoor *witness (i.e., witness for Item 9b) to finish this* WI *so that the simulator does not need to perform Stages 3, 5 and 7 honestly. Meanwhile, we will show that even a cheating P_2 cannot make the* trapdoor *witness available for her; So, she has to perform Stages 4, 6 and 8 honestly.*

10. **(P_2 Proves Honesty.)** P_1 and P_2 execute an instance of WI where P_2 proves that

(a) *either* it behaves honestly by following the instructions in Stages 4, 6 and 8; *or*

(b) there are more than $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ Stage 2 gadgets matching.

We remark that an honest P_2 will use the witness for Item 10a to finish this WI.

Comment. *This step is symmetric to Stage 10.*

Output: Both parties then output $r := r_1 \oplus r_2$ as the coin-flipping result.

Remark 12 (Potential Simplification of Prot. 5). We remark that Prot. 5 can be simplified. The security of it holds even if we remove Stage 6. The reason is: P_1 is the first party that reveals the random share r_1. In the case where P_1 is corrupted, the simulator can see this revealed r_1 before preparing the (simulated) share r_2. Thus, Stage 6 (which is designed for the extraction of r_1 when P_1 is corrupted) is not really necessary. But we choose to use the current version of Prot. 5 because it maintains symmetric roles for P_1 and P_2; This version generalizes to the multi-party setting (in Sec. 7) more easily.

Security Proof. The security of Prot. 5 is established by the following Thm. 9, whose proof is provided in Sec. 5.2.

Theorem 9. *For any $Q(\lambda)$ that is a polynomial of the security parameter λ, Prot. 5 is a Q-concurrent post-quantum two-party coin-flipping protocol.*

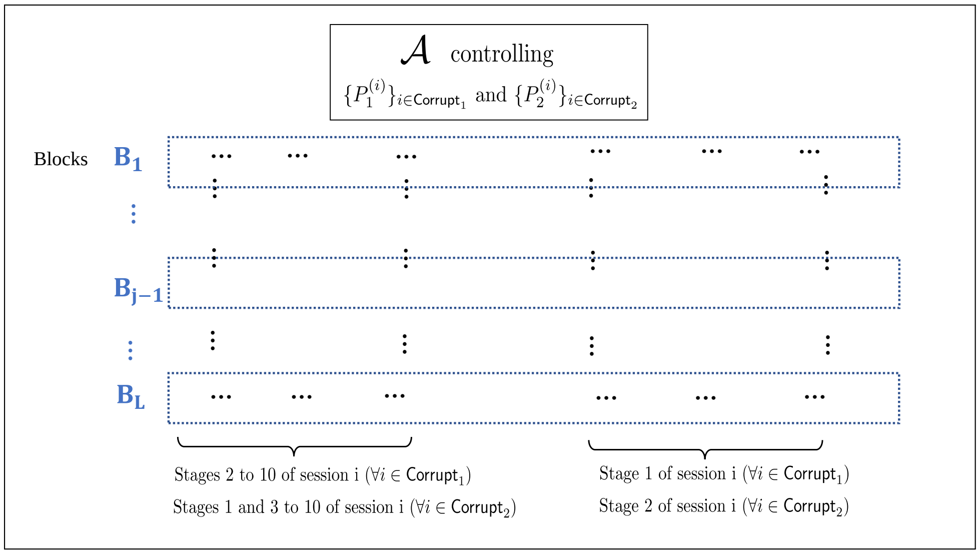
5.2 Security Proof (Proving Thm. 9)

In this section, we prove that Prot. 5 is secure in the Q-concurrent setting. We assume w.l.o.g. that the adversary corrupts exactly one party in each of the Q sessions. The sessions where both parties are corrupted or neither party is corrupted are not degenerated.

Recast the Execution to the MIM Setting. First, we recast the Q-concurrent execution of Prot. 5 into the main-in-the-middle setting. The purpose of this step is to present the schedule of messages in a similar pattern as the MIM execution of Q-concurrent simulation-sound ZK we discussed in Sec. 4, so that we can re-use the same proof technique as for the Q-concurrent simulation-sound of Prot. 4.

We first explain the intuition behind this recasting (to be specified formally later). Recall the Stage 2 of Prot. 4. In Prot. 4, let us call P the sender of Stage 2 and call V the receiver of Stage 2. Then, a feature of the Q-concurrent MIM execution of Prot. 4 is: On the left, $\mathcal{A}$ controls the receiver of the Stage 2 of each left session; On the right, $\mathcal{A}$ controls the sender of the Stage 2 in each right session. This structure played a central role when we prove Q-concurrent simulation soundness of Prot. 4. Now, we want to maintain this structure when recasting the Q-concurrent execution of Prot. 5 into the MIM setting. That is, we want to re-arrange the schedule of the Q-concurrent execution of Prot. 5 so that all the **Com-and-Guess Stages** where the receiver is corrupted appear on the left, and all the **Com-and-Guess Stages** where the sender is corrupted appear on the right.

This is a little more complex compared with the Q-concurrent MIM execution of Prot. 4. Because in *each* instance of Prot. 4 (in the Q-Q MIM setting), there is only *one* **Com-and-Guess Stage** (i.e., Stage 2) where



$\mathcal{A}$ controlling
$\{P_1^{(i)}\}_{i\in\mathsf{Corrupt}_1}$ and $\{P_2^{(i)}\}_{i\in\mathsf{Corrupt}_2}$
Blocks
$\mathbf{B_1}$
$\mathbf{B_{j-1}}$
$\mathbf{B_L}$
Stages 2 to 10 of session i ($\forall i \in \mathsf{Corrupt}_1$)
Stages 1 and 3 to 10 of session i ($\forall i \in \mathsf{Corrupt}_2$)
Stage 1 of session i ($\forall i \in \mathsf{Corrupt}_1$)
Stage 2 of session i ($\forall i \in \mathsf{Corrupt}_2$)

Fig. 3: Q-Concurrent Execution of Prot. 5 Recast to the MIM Setting

P is the sender and V is the receiver. In contrast, there are two **Com-and-Guess Stages** in Prot. 5 (i.e., Stages 1 and 2), where in Stage 1 P_1 is the sender, but in Stage 2, P_2 is the sender. Therefore, to maintain the aforementioned pattern, we do the following. For each session of Prot. 5 in the Q-concurrent execution, we put its **Com-and-Guess Stage** where the receiver is corrupted on the left, and put its **Com-and-Guess Stage** where the sender is corrupted on the right. Note that there are other stages apart from the two **Com-and-Guess Stages** in Prot. 5. They can be put on either side and it will not affect our analysis (roughly, that is because they have only constant rounds for each session). W.l.o.g., we choose to put them on the left.

We now formally describe the recast schedule. Let $\mathsf{Corrupt}_1$ be a subset of $[Q]$ such that for all sessions $i \in [Q]$, party $P_1^{(i)}$ is corrupted; Let $\mathsf{Corrupt}_2$ be a subset of $[Q]$ such that for all sessions $i \in [Q]$, party $P_2^{(i)}$ is corrupted. We first determine the messages that appear on the right. The right interaction consists of the following messages:

- The Stage 1 of session i for all $i \in \mathsf{Corrupt}_1$, *and*
- The Stage 2 of session i for all $i \in \mathsf{Corrupt}_2$.

All other stages will be put on the left. That is, the left interaction consists of the following messages

- The Stage 2 of session i for all $i \in \mathsf{Corrupt}_1$,
- The Stage 1 of session i for all $i \in \mathsf{Corrupt}_2$, *and*
- All other stages (i.e., except for Stages 1 and 2) of session i for all $i \in [Q]$.

This recast MIM schedule is depicted in Fig. 3. Finally, we highlight the following properties of it: (1) all the corrupted parties (controlled by $\mathcal{A}$) appears in the middle and all the left and right parties are honest; (2) all the **Com-and-Guess Stages** where the receiver is corrupted appear on the left; (3) all the **Com-and-Guess Stages** where the sender is corrupted appear on the right. Therefore, this schedule retains the structure of that of the Q-Q MIM execution of the simulation-sound ZK in Sec. 4. This is the reason why we can use a similar technique when proving security.

Number of Messages. We define some notations related to the round complexity of the execution shown in Fig. 3. Let T denote the total number of messages shown in Fig. 3. Since there are Q sessions in total, it follows that

$$T = Q \cdot \big(2 \cdot \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + 2 \cdot \varGamma_{\mathrm{SBC.C}} + 2 \cdot (\varGamma_{\mathrm{SFE}} + 1 + \varGamma_{\mathrm{WI}})\big), \tag{26}$$

where $s_{\mathsf{gad}} := \mathit{\Gamma}_{\mathrm{SBC}} + \mathit{\Gamma}_{\mathrm{ENMC}}$ is the number of messages of each **Com-and-Guess** gadget. Let T_{right} denote the number of messages shown on the right in Fig. 3. By the way we arrange the messages above, it follows that

$$T_{\mathsf{right}} = Q \cdot \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}}. \tag{27}$$

Each session $i \in [Q]$ has exactly

$$T^{(i)}_{\mathsf{right}} := \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} \tag{28}$$

messages on the left in Fig. 3.

Let T_{left} denote the number of messages shown on the left in Fig. 3. By the way we arrange the messages above, it follows that

$$T_{\mathsf{left}} = Q \cdot \big(\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + 2 \cdot \mathit{\Gamma}_{\mathrm{SBC.C}} + 2 \cdot (\mathit{\Gamma}_{\mathrm{SFE}} + 1 + \mathit{\Gamma}_{\mathrm{WI}})\big). \tag{29}$$

Each session $i \in [Q]$ has exactly

$$T^{(i)}_{\mathsf{left}} := \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + 2 \cdot \mathit{\Gamma}_{\mathrm{SBC.C}} + 2 \cdot (\mathit{\Gamma}_{\mathrm{SFE}} + 1 + \mathit{\Gamma}_{\mathrm{WI}}) \tag{30}$$

messages on the left in Fig. 3.

Intuition. Recall that we recast the execution to a similar structure as the case of simulation-sound ZK. Thus, the security proof follows from similar techniques used in Sec. 4.3. We will again partition these T messages (as shown in Fig. 3) into $L := 24Q^6\lambda$ equal-size blocks $\{B_1, \ldots, B_L\}$. That is, block B_1 contains the first $\frac{T}{L}$ messages, block B_2 contains the next $\frac{T}{L}$ messages, and so on (messages are ordered according to their order of appearing in the execution). We will refer to L as the *total number of blocks* and refer to $s_B := \frac{T}{L}$ as the *size* of each block. We will also define a sequence of hybrids indexed by $j \in [L]$, where roughly speaking, the j-th hybrid performs Watrous rewinding for the first $(j-1)$ blocks, but leaves the execution of the remaining blocks in straight-line. In the j-th hybrid, we check if any SBCom *given by a (simulated) honest party* (to his share of randomness) starts in the j-th block: If not, this hybrid is identical to the previous one; Otherwise, the (simulated) honest party will start to use a trapdoor extracted from the **Com-and-Guess Stage** to "cheat" in all his messages for the SBCom to his share, SFE, and WI. In this way, the simulator can extract the corrupted party's share and "enforce" the final coin-flipping result to the one from the ideal functionality.

Of course, we also need to ensure that the trapdoor witness is indeed available when a hybrid needs it, while the $\mathcal{A}$ cannot learn any trapdoor witness. Similar as in Sec. 4.3, this will be established with the help of a invariant condition which essentially says: the (simulated) honest parties can learn the trapdoor due to the performed Watrous rewinding, while the $\mathcal{A}$ cannot.[35] In the following, we first defined the invariant condition and then show the hybrids.

Definition 16 (Invariant Condition). *Let $\mathcal{T}$ be a transcript corresponding to an Q-concurrent execution of Prot. 5. Let* Corrupt$_1$ *and* Corrupt$_2$ *be as defined above. We say that* the invariant condition holds *if the following two requirements are satisfied:*

– *For all $i \in$ Corrupt$_1$ and $k \in [\ell_{\mathsf{gad}}]$, it holds that*

$$\Pr[a^{(i)}_{1,k} = b^{(i)}_{1,k}] = \frac{1}{2} \pm \mathsf{negl}(\lambda),$$

where $a^{(i)}_{1,k}$ is the value committed in the k-th ENMC *in Stage 1 of session i contained in $\mathcal{T}$, $b^{(i)}_{1,k}$ is the value committed in the k-th* SBCom *in Stage 1 of session i contained in $\mathcal{T}$, and the probability is taken over the random procedure generating $\mathcal{T}$.*

– *For all $i \in$ Corrupt$_2$ and $k \in [\ell_{\mathsf{gad}}]$, it holds that*

$$\Pr[a^{(i)}_{2,k} = b^{(i)}_{2,k}] = \frac{1}{2} \pm \mathsf{negl}(\lambda),$$

where $a^{(i)}_{2,k}$ is the value committed in the k-th ENMC *in Stage 2 of session i contained in $\mathcal{T}$, $b^{(i)}_{2,k}$ is the value committed in the k-th* SBCom *in Stage 2 of session i contained in $\mathcal{T}$, and the probability is taken over the random procedure generating $\mathcal{T}$.*

[35] This is essentially a "simulation-sound" type of requirement. That is why the current proof is highly similar to that in Sec. 4.3.

Hybrid REAL**:** This is the real Q-concurrent execution of Prot. 5 recast to the MIM execution as explain earlier (shown in Fig. 3). Let $\mathsf{OUT}(\mathrm{REAL})$ denote the output of the honest parties *and* the view of $\mathcal{A}$ in this game.

Lemma 12. *If* SBCom *is computationally hiding and* ENMC *is extractable (as per Def. 7), then the* invariant condition *(as per Def. 16) holds in hybrid* REAL*.*

Proof. This lemma is the counterpart of Lem. 8. It follows from the same argument as in the proof of Lem. 8.

Assume for contradiction that $\mathcal{A}$ manages to break the invariant condition for some $i \in [Q]$ and $k \in [\ell_{\mathsf{gad}}]$. If $i \in \mathsf{Corrupt}_1$ we can extract the $a^{(i)}_{1,k}$ (due to the extractability of ENMC) and break the computationally hiding property of SBCom using $a^{(i)}_{1,k}$ as a reasonable guess for the value $b^{(i)}_{1,k}$ committed in the corresponding SBCom. Similarly, if $i \in \mathsf{Corrupt}_2$ we can extract the $a^{(i)}_{2,k}$ and break the computationally hiding property of SBCom using $a^{(i)}_{2,k}$ as a reasonable guess for the value $b^{(i)}_{2,k}$ committed in the corresponding SBCom. This is exactly the same argument that we used in the proof of Lem. 8 (which is also used in Sec. 4.2). We omit the details. □

Hybrid $G_{0,6}$**:** This hybrid is the counterpart of the $H_{0,2}$ defined on Page 25. It sets up necessary quantum registers and performs the REAL execution *in a coherent manner*. By comparing with the $H_{0,2}$ on Page 25, it is straightforward to see how this hybrid should be defined. The only difference that we want to high-light is how the Watrous control registers $\otimes_{j=1}^{L}\mathtt{W}_j$ are set:

– $\otimes_{j=1}^{L}\mathtt{W}_j$: Similar as in the $H_{0,2}$ defined on Page 25, we need L Watrous control registers $\otimes_{j=1}^{L}\mathtt{W}_j$ to control the rewinding behavior for each block. Formally, for all $j \in [L]$, each $\mathtt{W}_j$ is set using the $\mathsf{Modify}(\mathtt{W}_j)$ algorithm shown in Algo. 5.1.

Algorithm 5.1: $\mathsf{Modify}(\mathtt{W}_j)$

For all $i \in [Q]$ and $j \in L$, let $K^{(i)}_j$ be a subset of $[\ell_{\mathsf{gad}}]$ defined as follows:

– If $i \in \mathsf{Corrupt}_1$, then $k \in K^{(i)}_j$ if and only if the k-th Stage 2 gadget of session i is fully nested in block B_j.

– If $i \in \mathsf{Corrupt}_2$, then $k \in K^{(i)}_j$ if and only if the k-th Stage 1 gadget of session i is fully nested in block B_j.

For all $j \in [L]$, let T_j be a subset of $[Q]$ such that $i \in T_j$ if and only if $K^{(i)}_j \neq \emptyset$.
Then, for all $j \in L$, the Watrous control register $\mathtt{W}_j$ is set to a binary bit c_j defined as follows:

– If $T_j = \emptyset$, then sample $c_j \xleftarrow{\$} \{0,1\}$ uniformly at random;

– Otherwise (i.e., $T_j \neq \emptyset$), sample $i \xleftarrow{\$} T_j$ and then sample $k \xleftarrow{\$} K^{(i)}_j$. In this case, c_j is defined as follows according to the sampled i and k:

- If $i \in \mathsf{Corrupt}_1$, then $c_j = 1$ if and only if $a^{(i)}_{2,k} \neq b^{(i)}_{2,k}$, where $a^{(i)}_{2,k}$ is the value committed in the ENMC in the k-th Stage 2 gadget of session i, and $b^{(i)}_{2,k}$ is the value committed in the SBCom in the k-th Stage 2 gadget of session i;
- If $i \in \mathsf{Corrupt}_2$, then $c_j = 1$ if and only if $a^{(i)}_{1,k} \neq b^{(i)}_{1,k}$, where $a^{(i)}_{1,k}$ is the value committed in the ENMC in the k-th Stage 1 gadget of session i, and $b^{(i)}_{1,k}$ is the value committed in the SBCom in the k-th Stage 1 gadget of session i.

Comment. *This algorithm should be interpreted as follows. In each block B_j, it checks if there exist any fully nested* gadgets *where the* receiver *is corrupted (i.e., a* gadget *on the left in Fig. 3)—That is, if $P^{(i)}_1$ (resp. $P^{(i)}_2$) is corrupted, then it checks if there exist any fully nested Stage 2 (resp. Stage 1)* gadgets *of session i. If so, it samples at random a fully nested* gadget *(by $i \xleftarrow{\$} T_j$ and $k \xleftarrow{\$} K^{(i)}_j$) and sets $c_j = 1$ if and only if this sampled* gadget *does not match. If not (i.e., there is no fully nested* gadgets *in B_j where the* receiver *is corrupted), it sets c_j to 1 with probability 1/2.*

Lemma 13. *It holds that* $\mathsf{OUT}(\text{REAL}) \overset{\text{i.d.}}{=\!=} \mathsf{OUT}(G_{0,6})$*. Moreover, if the* invariant condition *(as per Def. 16) holds in* REAL*, it must hold in* $G_{0,6}$ *as well.*

Proof. This lemma is the counterpart of Lem. 9. It follows from exactly the same proof techniques. We omit the details. □

Comment. *In the following, we define the hybrids* $\{G_{j,1}, \ldots, G_{j,5}\}$*. These five hybrids* together *can be viewed as the counterpart of the* $H_{j,1}$ *defined on Page 27. One may wonder why the* $H_{j,1}$ *has five (instead of just one) counterparts in this proof. The reason is: In the proof shown in Sec. 4.3, there is only one component for which we want to get rid of the real input (i.e., the witness* w *in the ZK setting), which is the* WI*; However, in the current proof, the counterparts of this component include the* SBCom *to share of randomness,* SFE*, and the* WI *messages given by the (simulated) honest party. The simulator will consume one hybrid to "cheat" in one of them. See the following for details.*

Hybrid $G_{j,1}$ **(**$\forall j \in [L]$**):** This hybrid is the counterpart of the $H_{j,1}$ defined on Page 27. It is identical to $G_{j-1,6}$ except that $G_{j,1}$ starts to use the trapdoor witness in the WI where the verifier is corrupted. In more detail:

- For all session i where $P_1^{(i)}$ is corrupted, if its Stage 4 SBCom starts in block B_j, then $P_2^{(i)}$ starts to use the trapdoor witness (i.e., there are more than $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ Stage 2 gadget matching) in the Stage 10 WI.
- For any session i where $P_2^{(i)}$ is corrupted, if its Stage 3 SBCom starts in block B_j, then $P_1^{(i)}$ starts to use the trapdoor witness (i.e., there are more than $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ Stage 1 gadget matching) in the Stage 9 WI.

Similar as in $H_{j,1}$, it is possible that when the $P_1^{(i)}$ (or $P_2^{(i)}$) needs to use the trapdoor witness, this trapdoor witness is not available. In that case, $G_{j,1}$ halts immediately and outputs a special symbol $\mathsf{Abort}_{\text{NT}}$. We denote this event by E_{NT}. (Looking ahead, we will prove in Lem. 14 that this happens with negligible probability.) This finishes the description of $G_{j,1}$.

Lemma 14. *If* WI *is witness indistinguishable, then for any* $j \in [L]$*, it holds that* $\mathsf{OUT}(G_{j-1,6}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{j,1})$*. Moreover, if* SBCom *is computationally hiding,* ENMC *is both extractable (as per Def. 7) and first-message binding (as per Rmk. 4), and that the* invariant condition *holds in* $G_{j-1,6}$ $(\forall j \in [L])$*, then the* invariant condition *hold in* $G_{j,1}$ $(\forall j \in [L])$ *as well.*

Proof. This lemma can be proven following the same argument as for Lem. 10.

Proving Indistinguishability. To prove $\mathsf{OUT}(G_{j-1,6}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{j,1})$, notice that hybrids $G_{j-1,6}$ and $G_{j,1}$ are identical until the end of block B_{j-1}. They start to differ only if for some $i \in [Q]$, the SBCom (given by the uncorrupted party) of session i starts in block B_j. If that happens, $G_{j-1,6}$ proceeds using the real witness, but $G_{j,1}$ proceeds using the trapdoor witness (*if the* trapdoor *witness is available*). Also note that starting from block B_j, the execution in both hybrids are straight-line. So the indistinguishability follows from the WI property of this WI. Thus, to finish this proof, the only thing we need to prove is the following Claim 10, which can be understood as the counterpart of Claim 5.

Claim 10. *It holds that* $\Pr_{G_{j,1}}[E_{\text{NT}}] \le \mathsf{negl}(\lambda)$*.*

Proof. This proof is almost identical to that of Claim 5. We present it in Sec. 5.3. □

Proving Invariant Condition. The invariant condition can be proven using exactly the same technique as in Sec. 4.4.2. Consider the invariant condition for the k-th gadget in Stage 1 of session $i \in \mathsf{Corrupt}_1$. We use the final message of block B_{j-1} as a pivot to divide the schedule into the following four cases (similar as those shown in Fig. 2) and establish the invariant condition for each them separately:

1. The decommitment to $b_{1,k}^{(i)}$ ends before (or in parallel with) the final message of B_{j-1}.
2. The SBCom to $b_{1,k}^{(i)}$ starts after the final message of B_{j-1};

3. The SBCom to $b^{(i)}_{1,k}$ ends before (or in parallel with) the final message of B_{j-1}, but the ENMC to $a^{(i)}_{1,k}$ starts after the final message of B_{j-1};
4. The ENMC to $a^{(i)}_{1,k}$ starts before (or in parallel with) the final message of B_{j-1}, but this ENMC *but ends after* final message of B_{j-1}.

The invariant condition for each of the above four cases follows from exactly the same argument as shown on Page 30. We thus omit the details.

The invariant condition for the k-th gadget in Stage 2 of session $i \in \mathsf{Corrupt}_2$ can be proven similarly. The only difference is: instead of the $a^{(i)}_{1,k}$ and $b^{(i)}_{1,k}$ in the Stage 1 gadgets, we will focus on the $a^{(i)}_{2,k}$ and $b^{(i)}_{2,k}$ in the Stage 2 gadgets to perform the above proof.

This finishes the proof of Lem. 14. □

Hybrid $G_{j,2}$ ($\forall j \in [L]$): This hybrid is identical to $G_{j,1}$ except that $G_{j,2}$ makes the SFE stage independent of honest parties' committed share of randomness. In more detail:

– For any session i where $P^{(i)}_1$ is corrupted, if its Stage 4 SBCom starts in block B_j, then $P^{(i)}_2$ finishes this Stage 5 SFE using a different circuit C'_1 (in place of C_1) defined as follows:
  - C'_1: it is a dummy circuit that always outputs the symbol $\vdash$, but padded to the same topology and size of the C_1 defined in Stage 5. Importantly, C'_1 does not have r_2 hard-wired any more.

  We remark that all other steps are performed in exactly the same manner as in $G_{j,1}$. The only thing that changes is the definition of the circuit (from C_1 to C'_1) in this SFE.

– For any session i where $P^{(i)}_2$ is corrupted, if its Stage 3 SBCom starts in block B_j, then $P^{(i)}_1$ finishes this Stage 6 SFE using a different circuit C'_2 (in place of C_2) defined as follows:
  - C'_2: it is a dummy circuit that always outputs the symbol $\vdash$, but padded to the same topology and size of the C_2 defined in Stage 6. Importantly, C'_2 does not have r_1 hard-wired any more.

  We remark that all other steps are performed in exactly the same manner as in $G_{j,1}$. The only thing that changes is the definition of the circuit (from C_2 to C'_2) in this SFE.

This finishes the description of $G_{j,2}$.

Lemma 15. *If* SFE *is function hiding (as per Lem. 5), then for any $j \in [L]$, it holds that* $\mathsf{OUT}(G_{j,1}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{j,2})$. *Moreover, if* SBCom *is computationally hiding,* ENMC *is both extractable (as per Def. 7) and first-message binding (as per Rmk. 4), and that the* invariant condition *holds in $G_{j,1}$ ($\forall j \in [L]$), then the* invariant condition *hold in $G_{j,2}$ ($\forall j \in [L]$) as well.*

Proof. The proof of this lemma is almost identical to that of Lem. 14. The only difference is: when proving $\mathsf{OUT}(G_{j,1}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{j,2})$, we rely on the function hiding property of SFE (instead of the WI property of WI as in the proof of Lem. 14).

One caveat is that we need to make sure that the C'_1 (resp. C'_2) should be functionally identical to C_1 (resp. C_2) w.r.t. $\mathcal{A}$. That is, $\mathcal{A}$ cannot find an input that distinguishes the original circuit and the prime'd version (except for with negligible probability). To see that, recall that C'_1 and C_1 (resp. C'_2 and C_2) differ only if the input is a valid trapdoor witness. Since the trapdoor witness is not available to $\mathcal{A}$ in the concerned session (due to the invariant condition, which could be proven using exactly the same argument as in Lem. 14.), the above claim holds. We omit the details. □

Hybrid $G_{j,3}$ ($\forall j \in [L]$): This hybrid is identical to $G_{j,2}$ except that $G_{j,3}$ makes the honest parties commit to an all-0 string as the share of randomness. In more detail:

– For any session i where $P^{(i)}_1$ is corrupted, if its Stage 4 SBCom starts in block B_j, then $P^{(i)}_2$ finishes this SBCom by committing to an all-0 string of the same length of r_2.

– For any session i where $P^{(i)}_2$ is corrupted, if its Stage 3 SBCom starts in block B_j, then $P^{(i)}_1$ finishes this SBCom by committing to an all-0 string of the same length of r_1.

This finishes the description of $G_{j,3}$.

Lemma 16. *If* SBCom *is computationally hiding, then for any* $j \in [L]$*, it holds that* $\mathsf{OUT}(G_{j,2}) \overset{c}{\approx} \mathsf{OUT}(G_{j,3})$*. Moreover, if* ENMC *is both extractable (as per Def. 7) and first-message binding (as per Rmk. 4), and that the* invariant condition *holds in* $G_{j,2}$ $(\forall j \in [L])$*, then the* invariant condition *hold in* $G_{j,3}$ $(\forall j \in [L])$ *as well.*

Proof. The proof of this lemma is almost identical to that of Lem. 14. The only difference is: when proving $\mathsf{OUT}(G_{j,2}) \overset{c}{\approx} \mathsf{OUT}(G_{j,3})$, we rely on the computationally hiding property of SBCom (instead of the WI property of WI as in the proof of Lem. 14). We omit the details. □

Hybrid $G_{j,4}$ $(\forall j \in [L])$**:** This hybrid is identical to $G_{j,3}$ except that $G_{j,4}$ tries to learn $\mathcal{A}$'s share of randomness from the SFE stage. In more detail:

- For any session i where $P_1^{(i)}$ is corrupted, if its Stage 4 SBCom starts in block B_j, then $P_2^{(i)}$ finishes the Stage 6 SFE in the following manner. It sets $w_2^{(i)}$ to the $\mathsf{trapdoor}$ witness; When it receives $\widehat{\mathsf{ct}}_2^{(i)}$ from the corrupted $P_1^{(i)}$, it computes $r_1^{(i)} = \mathsf{SFE.Dec}(\mathsf{k}_2^{(i)}, \widehat{\mathsf{ct}}_2^{(i)})$.
- For any session i where $P_2^{(i)}$ is corrupted, if its Stage 3 SBCom starts in block B_j, then $P_1^{(i)}$ finishes the Stage 5 SFE in the following manner. It sets $w_1^{(i)}$ to the $\mathsf{trapdoor}$ witness; When it receives $\widehat{\mathsf{ct}}_1^{(i)}$ from the corrupted $P_2^{(i)}$, it computes $r_2^{(i)} = \mathsf{SFE.Dec}(\mathsf{k}_1^{(i)}, \widehat{\mathsf{ct}}_1^{(i)})$.

This finishes the description of $G_{j,4}$.

Lemma 17. *If* SFE *is input hiding (as per Property 2 in Def. 9), then for any* $j \in [L]$*, it holds that* $\mathsf{OUT}(G_{j,3}) \overset{c}{\approx} \mathsf{OUT}(G_{j,4})$*. Moreover, if* SBCom *is computationally hiding,* ENMC *is both extractable (as per Def. 7) and first-message binding (as per Rmk. 4), and that the* invariant condition *holds in* $G_{j,3}$ $(\forall j \in [L])$*, then the* invariant condition *hold in* $G_{j,4}$ $(\forall j \in [L])$ *as well.*

Proof. The proof of this lemma is almost identical to that of Lem. 14. The only difference is: when proving $\mathsf{OUT}(G_{j,3}) \overset{c}{\approx} \mathsf{OUT}(G_{j,4})$, we rely on the input hiding property of SFE (instead of the WI property of WI as in the proof of Lem. 14). We omit the details. □

Hybrid $G_{j,5}$ $(\forall j \in [L])$**:** This hybrid is identical to $G_{j,4}$ except that $G_{j,5}$ starts to "force" the coin-flipping result to the one from the ideal coin-flipping functionality, by making use of the adversary's share of randomness extracted from hybrid $G_{j,4}$. In more detail:

- For any session i where $P_1^{(i)}$ is corrupted, if its Stage 4 SBCom starts in block B_j, $P_2^{(i)}$ sends $r_2^{(i)} \coloneqq r^{(i)} \oplus r_1^{(i)}$ in Stage 8, where the $r^{(i)}$ is the coin-flipping result from the ideal functionality (see also Rmk. 13) and $r_1^{(i)}$ is the corrupted $P_1^{(i)}$'s share extracted from hybrid $G_{j,4}$.
- For any session i where $P_2^{(i)}$ is corrupted, if its Stage 3 SBCom starts in block B_j, $P_1^{(i)}$ sends $r_1^{(i)} \coloneqq r^{(i)} \oplus r_2^{(i)}$ in Stage 7, where the $r^{(i)}$ is the coin-flipping result from the ideal functionality (see also Rmk. 13) and $r_2^{(i)}$ is the corrupted $P_2^{(i)}$'s share extracted from hybrid $G_{j,4}$.

This finishes the description of $G_{j,5}$.

Remark 13. Note that starting from $G_{1,5}$, we start to use the ideal coin-flipping functionality $\mathcal{F}_{\text{CP}}$, which does not take any input but output a random string $r^{(i)}$ to the parties of the i-th session, where i is as described above (i.e., the sessions i where $P_1^{(i)}$ is corrupted and the Stage 4 SBCom starts in block B_j, or $P_2^{(i)}$ is corrupted and the Stage 3 SBCom starts in block B_j). Let us call these sessions as the concerned session i.

Lemma 18. *For any* $j \in [L]$*, it holds that* $\mathsf{OUT}(G_{j,4}) \overset{i.d.}{=\!=} \mathsf{OUT}(G_{j,5})$*. Moreover, if the* invariant condition *holds in* $G_{j,4}$ $(\forall j \in [L])$*, then it holds in* $G_{j,5}$ $(\forall j \in [L])$ *as well. Also, if the* WI *is computationally sound, then for the* concerned session i *described above, the output of* $P_1^{(i)}$ *and* $P_2^{(i)}$ *will be the* $r^{(i)}$ *from the ideal* $\mathcal{F}_{\text{CP}}$ *as described in Rmk. 13.*

Proof. Both the view indistinguishability and $\mathsf{invariant\ condition}$ follows from the fact that $\mathcal{A}$'s view in $G_{j,5}$ are identically distributed to that in $G_{j,4}$

Also, for the $\mathsf{concerned\ session}$ i, it is straightforward that the coin-flipping result will be "forced" to the $r^{(i)}$ from $\mathcal{F}_{\text{CP}}$, as long as the corrupted party, say $P_b^{(i)}$, cannot make the share $r_b^{(i)}$ hard-wired in the SFE circuit *inconsistent* with the $r_b^{(i)}$ committed in the SBCom. To prove this, notice that the $\mathsf{trapdoor}$ witness is not available to this corrupted $P_b^{(i)}$ due to the $\mathsf{invariant\ condition}$. Therefore, the consistency between SFE and SBCom follows immediately from the soundness of WI. □

Hybrid $G_{j,6}$ ($\forall j \in [L]$): This hybrid is the counterpart of the $H_{j,2}$ defined on Page 27. That is, $G_{j,6}$ finishes block B_j using Watrous rewinding, with $\mathtt{W}_j$ playing the role as the $\mathsf{Watrous\ control\ register}$. All other steps are performed in exactly the same manner as in $G_{j,5}$. This finishes the description of $G_{j,6}$.

Lemma 19. *If* ENMC *is computationally hiding and* WI *is witness indistinguishable, then for any* $j \in [L]$, *it holds that* $\mathsf{OUT}(G_{j,5}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{j,6})$. *Moreover, if we additionally assume that* ENMC *is both non-malleable and first-message binding (as per Rmk. 4), and the* $\mathsf{invariant\ condition}$ *holds in* $G_{j,5}$ $(\forall j \in [L])$, *then the* $\mathsf{invariant\ condition}$ *hold in* $G_{j,6}$ $(\forall j \in [L])$ *as well.*

Proof. This proof is identical to that of Lem. 11 (shown in Sec. 4.5). We omit the details. □

Finishing the Proof. The final simulator $\mathcal{S}$ in the Q-concurrent execution can be constructed in the following way:

– $\mathcal{S}$ simply emulates the hybrid $G_{L,6}$. Note that $G_{L,6}$ needs to access $\mathcal{A}$, which $\mathcal{S}$ will provide because $\mathcal{S}$ has access to $\mathcal{A}$ for simulation; Also, $G_{L,6}$ needs to access the ideal functionality $\mathcal{F}_{\text{CP}}$, which will also be through $\mathcal{S}$, who will simply forward the output of $\mathcal{F}_{\text{CP}}$ to $G_{L,6}$.

It follows from Lem. 12 to Lem. 19 that $\mathsf{OUT}(\text{REAL}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{L,6})$, which further implies that the ideal-world Q-concurrent execution simulated by $\mathcal{S}$ is computationally indistinguishable to that in the real world.

This finishes the proof of Thm. 9.

5.3 Proof of Claim 10

Intuition. This claim follows from the same technique as for the proof of Claim 5 shown in Appx. B.

At a high-level, for any session $i \in [Q]$, we again divide all the its $\mathsf{gadgets}$ appearing on the left (in Fig. 3) into two types: (1) matching by rigging, and (2) matching by luck, following the same definition as in Appx. B. We will then argue that with overwhelming probability, the total number of these two types of $\mathsf{gadgets}$ exceeds the threshold $\mathsf{Th} \coloneqq 60Q^7\lambda + Q^4\lambda$, which implies that the $\mathsf{trapdoor}$ witness in session i is available to the uncorrupted party (i.e., the role played by the simulator) in this session.

Almost all the steps are identical as in Appx. B. It is only the derivation of the bounds for Rig_i and Luck_i that will be slightly different. But even for this step, the eventual bounds we obtain are still identical to those in Appx. B. Therefore, we only need to focus on this derivation in the following.

Notation. Similar as in Appx. B, we use Rig_i to denote the number of left $\mathsf{gadgets}$ of session i that matches because of the Watrous rewinding in hybrid $G_{j,1}$, and use Luck_i to denote the number of left $\mathsf{gadgets}$ of session i that matches because of pure luck in hybrid $G_{j,1}$. Again, our actual proof will only focus on the Rig'_i and Luck'_i, which are the counterparts of Rig_i and Luck_i but in the REAL game, because it can be shown (following exactly the same argument in Sec. B.1) that bounding Rig'_i and Luck'_i already suffices for the current proof.

In the following, we first prove a claim (Claim 11) that can be considered as a counterpart of Claim 19. Then, we proceed to define Rig'_i and Luck'_i formally, and show lower-bounds for them.

For a session where P_1 (resp. P_2) is corrupted, we first bound (in Claim 11) the max number of its Stage 1 (resp. Stage 2) $\mathsf{gadgets}$ that can be fully nested in a block. Claim 11 is the counterpart of Claim 19.

Claim 11. *For any session* $i \in \mathsf{Corrupt}_1$, *it can have at most* $12Q^2$ *Stage 2* $\mathsf{gadgets}$ *fully nested in any single block. For session* $i \in \mathsf{Corrupt}_2$, *it can have at most* $12Q^2$ *Stage 1* $\mathsf{gadgets}$ *fully nested in any single block.*

Proof. Recall from our parameter setting that $\ell_{\mathsf{gad}} = 120Q^7\lambda$ and $L = 24Q^6\lambda$. Also recall from Eq. (26) that there are

$$T = Q \cdot \big(2 \cdot \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + 2 \cdot \mathit{\Gamma}_{\text{SBC.C}} + 2 \cdot (\mathit{\Gamma}_{\text{SFE}} + 1 + \mathit{\Gamma}_{\text{WI}})\big)$$

messages in the Q-concurrent MIM execution of Prot. 5, where $s_{\mathsf{gad}} = \mathit{\Gamma}_{\text{SBC}} + \mathit{\Gamma}_{\text{ENMC}}$ denotes the size of each gadget. If we let s_B denote the size of each block, then it holds that

$$s_B = \frac{T}{L} = 10Q^2 \cdot s_{\mathsf{gad}} + \frac{C}{24Q^5\lambda} \leq 12Q^2 \cdot s_{\mathsf{gad}},$$

where $C := 2 \cdot \mathit{\Gamma}_{\text{SBC.C}} + 2 \cdot (\mathit{\Gamma}_{\text{SFE}} + 1 + \mathit{\Gamma}_{\text{WI}})$ is a *constant.*

This implies that each block can at most contain $12Q^2$ fully nested $\mathsf{gadgets}$, which further implies Claim 11. □

Defining Rig'_i and Luck'_i. Similar as in Algo. B.2, we first define the random variables $\mathit{\Sigma}'$, $X'_{i,j}$, $\mathit{\Theta}'_i$, and $Z'_{i,k}$ in REAL. Recall that the current hybrid is $G_{j,1}$. Similar as in Algo. B.2, we writes this j as j^* in the following to emphasize this index and to avoid potential index conflicts.

Algorithm 5.2: Procedure Pick regarding REAL

Execute the REAL game. This yields a schedule S. Let $\{B_1, \ldots, B_L\}$ be the partition associated with S. For this S, consider the following sub-procedure:

Procedure $\mathsf{SubPick}(S)$. Iterate for $j = 1$ to $j^* - 1$:

1. Let T_j and $K_j^{(i)}$ ($\forall i \in [Q]$) be defined as in Algo. 5.1;
2. Sample $i \xleftarrow{\$} T_j$; It is possible that $T_j = \emptyset$. In that case, directly move onto the next iteration.
3. Sample $k \xleftarrow{\$} K_j^{(i)}$.

Random Variables: We define some random variables w.r.t. the above random procedure:

– Let $\mathit{\Sigma}'$ denote the subset of $[Q]$ such that $i \in \mathit{\Sigma}'$ iff the **Stage-$*$** SBCom of session i starts in block B_{j^*}. (The "**Stage-$*$**" will be determined depending on which party is corrupted in session i—If $i \in \mathsf{Corrupt}_1$, then **Stage-$*$** stands for Stage 4; If $i \in \mathsf{Corrupt}_2$, then **Stage-$*$** stands for Stage 3.)

– For any $i \in \mathit{\Sigma}'$ and any $j \in [j^* - 1]$, let $X'_{i,j}$ be a binary random variable defined to be 1 iff in Step 2 of the j the iteration of the above $\mathsf{SubPick}(S)$ procedure, the index i is sampled.

– For any $i \in \mathit{\Sigma}'$, let $\mathit{\Theta}'_i$ denote the indices of the *first* $3Q^4\lambda$ **Stage-$*$** $\mathsf{gadgets}$ that are sampled by the above procedure in session i. (The "**Stage-$*$**" will be determined depending on which party is corrupted in session i—If $i \in \mathsf{Corrupt}_1$, then **Stage-$*$** stands for Stage 2; If $i \in \mathsf{Corrupt}_2$, then **Stage-$*$** stands for Stage 1.)

– For any $i \in \mathit{\Sigma}'$ and any $k \in [\ell_{\mathsf{gad}}] \setminus \mathit{\Theta}'_i$, let $Z'_{i,k}$ be a binary random variable defined as follows:
  - **(Type-1 k):** If this k-th **Stage-$*$** gadget of session i is sampled by the above procedure, let $Z'_{i,k}$ be a Bernoulli random variable with $p = \frac{1}{2}$;
  - **(Type-2 k):** Otherwise, let $Z'_{i,k} = 1$ iff the k-th **Stage-$*$** gadget of session i matches,

  where, again, the "**Stage-$*$**" will be determined by which party is corrupted in session i—If $i \in \mathsf{Corrupt}_1$, then **Stage-$*$** stands for Stage 2; If $i \in \mathsf{Corrupt}_2$, then **Stage-$*$** stands for Stage 1.

The Rig'_i and Luck'_i are defined in exactly the same way as on Page 69. That is, we define

$$\forall i \in \mathit{\Sigma}',\ \mathsf{Rig}'_i := \begin{cases} 3Q^4\lambda & \text{if } \sum_{j\in[j^*-1]} X'_{i,j} \geq 3Q^4\lambda \\ \sum_{j\in[j^*-1]} X'_{i,j} & \text{if } \sum_{j\in[j^*-1]} X'_{i,j} < 3Q^4\lambda \end{cases}, \tag{31}$$

and

$$\forall i \in \mathit{\Sigma}',\ \mathsf{Luck}'_i := \sum_{k\in[\ell_{\mathsf{gad}}]\setminus\mathit{\Theta}'_i} Z'_{i,k}, \tag{32}$$

where the probability is taken over the whole random procedure Pick shown by Algo. 5.2.

Bounding Rig'_i and Luck'_i. To bound Rig'_i and Luck'_i, it suffices to show the counterparts of Claims 21 and 22. The remainder of this proof is devoted to that.

We first prove the following Claim 12, which is the counterpart of Claim 20.

Claim 12. *Fix an S in the support of Pick shown in Algo. 5.2. For any $i \in [Q]$, we define N_i as follows:*

– *If $i \in \mathsf{Corrupt}_1$, N_i denotes the number of blocks that have at least one fully nested Stage 2 gadget of the session i in S.*

– *If $i \in \mathsf{Corrupt}_2$, N_i denotes the number of blocks that have at least one fully nested Stage 1 gadget of the session i in S.*

It holds that for all $i \in [Q]$, $N_i \geq 6Q^5\lambda - \mathsf{const}$, where

$$\mathsf{const} := 2 \cdot \mathit{\Gamma}_{\text{SBC.C}} + 2 \cdot (\mathit{\Gamma}_{\text{SFE}} + 1 + \mathit{\Gamma}_{\text{WI}})$$

is a constant.

Proof. This proof follows the same argument as for Claim 20.

Note that if $i \in \mathsf{Corrupt}_1$, the Stage 2 $\mathsf{gadgets}$ of the session i in S appear on the left in the MIM execution (shown in Fig. 3); if $i \in \mathsf{Corrupt}_2$, the Stage 1 $\mathsf{gadgets}$ of the session i in S also appear on the left in the MIM execution (shown in Fig. 3). Thus, our proof actually does not need to distinguish between the case $i \in \mathsf{Corrupt}_1$ and the case $i \in \mathsf{Corrupt}_2$. We just need to focus on the **Com-and-Guess Stages** that appear on the left.

For any $i \in [Q]$, we call the messages appearing on the left side in Fig. 3 as *the left messages of session i*. Recall from Eq. (30) that there are $T^{(i)}_{\mathsf{left}} = \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const}$ left messages of session i in total, where

$$\mathsf{const} := 2 \cdot \mathit{\Gamma}_{\text{SBC.C}} + 2 \cdot (\mathit{\Gamma}_{\text{SFE}} + 1 + \mathit{\Gamma}_{\text{WI}})$$

Let u_i denote the number of blocks that contain at least $2s_{\mathsf{gad}}$ left messages of session i (recall that s_{gad} is the size of each gadget). Note that $(u_i - \mathsf{const})$ lower-bounds N_i.

Let $\{b_1, \ldots, b_{u_i}\}$ be the number of left messages of session i that are contained in each of these u_i blocks. Let the number of left messages of session i contained in the remaining $L - u_i$ blocks be denoted as $\{a_1, \ldots, a_{L-u_i}\}$. Notice that by definition, each a_k ($\forall k \in [L - u_i]$) block contains $< 2s_{\mathsf{gad}}$ left messages of session i. Thus, it holds that

$$\sum_{k=1}^{L-u_i} a_k < (L - u_i) \cdot 2s_{\mathsf{gad}}. \tag{33}$$

Also, since the total number of messages contained in each block is s_B, it holds that

$$\sum_{k=1}^{u_i} b_k \leq u_i \cdot s_B. \tag{34}$$

Also recall s_B is equal to the total number of messages in the Q-concurrent execution divided by the number of blocks L. That is,

$$s_B = \frac{Q \cdot \big(2 \cdot \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + 2 \cdot \mathit{\Gamma}_{\text{SBC.C}} + 2 \cdot (\mathit{\Gamma}_{\text{SFE}} + 1 + \mathit{\Gamma}_{\text{WI}})\big)}{L} = \frac{2Q(\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const}')}{L}, \tag{35}$$

where $\mathsf{const}' := \frac{\mathsf{const}}{2}$.

Observe that the sum of these b_k's and a_k's is exactly the total number of the left messages of session i, which is exactly $T^{(i)}_{\mathsf{left}}$. That is,

$$\sum_{k=1}^{u_i} b_k + \sum_{k=1}^{L-u_i} a_k = T^{(i)}_{\mathsf{left}} = \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const}. \tag{36}$$

Eq. (36) together with Inequalities (33) and (34) imply that:

$$u_i \cdot s_B + (L - u_i) \cdot 2s_{\mathsf{gad}} \geq \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const},$$

which further implies that:

$$\begin{aligned}
u_i &\geq \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const} - 2s_{\mathsf{gad}} L}{s_B - 2s_{\mathsf{gad}}} \\
&= \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const} - 2s_{\mathsf{gad}} L}{\frac{2Q(\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const}')}{L} - 2s_{\mathsf{gad}}} && (37) \\
&= \frac{L}{2Q} \cdot \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const} - 2s_{\mathsf{gad}} L}{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const}' - 2s_{\mathsf{gad}} \cdot \frac{L}{2Q}} \\
&> \frac{L}{2Q} \cdot \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const} - 2s_{\mathsf{gad}} L}{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const} - 2s_{\mathsf{gad}} \cdot \frac{L}{2Q}} && (38) \\
&\geq \frac{L}{2Q} \cdot \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const} - 2s_{\mathsf{gad}} L}{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const}} \\
&= \frac{L}{2Q} \cdot \left(1 - \frac{2s_{\mathsf{gad}} L}{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathsf{const}}\right) \\
&\geq \frac{L}{2Q} \cdot \left(1 - \frac{2L}{\ell_{\mathsf{gad}}}\right) \\
&= \frac{L}{2Q} \cdot \left(1 - \frac{2}{5Q}\right) && (39) \\
&\geq \frac{L}{2Q} \cdot \left(1 - \frac{1}{2}\right) \\
&= 6Q^5\lambda && (40)
\end{aligned}$$

where Eq. (37) follows Eq. (35), Eq. (38) follows from the fact that $\mathsf{const}' < \mathsf{const}$, Eq. (39) follows from our parameter setting of $L = 24Q^6\lambda$ and $\ell_{\mathsf{gad}} = 120Q^7\lambda$, and Eq. (40) follows from our parameter setting of $L = 24Q^6\lambda$.

As mentioned before, $u_i - \mathsf{const}$ lower-bounds N_i. It then follows from Eq. (40) that $N_i \geq 6Q^5\lambda - \mathsf{const}$. This finishes the proof of Claim 12.

□

Next, we show the following Claim 13, which is a counterpart of Claim 21.

Claim 13 (Matching by Rigging)**.** *It holds that*

$$\Pr\left[\exists i \in \Sigma' \text{ s.t. } \sum_{j \in [j^*-1]} X'_{i,j} < 3Q^4\lambda\right] \leq \mathsf{negl}(\lambda), \qquad (41)$$

where both the expectation and the probability is taken over the whole random procedure Pick *defined by Algo. 5.2.*

Proof. This proof is identical to that of Claim 21. Following the same argument, one can show the following counterpart of Inequality (64) (which makes use of Claim 12):

$$\forall i \in \sigma',\ \mathbb{E}\left[\textstyle\sum_{j \in [j^*-1]} X'_{i,j}\right] \geq 6Q^4\lambda - \tfrac{\mathsf{const}}{Q} - 12Q, \qquad (42)$$

where σ' is an instantiation of Σ' resulted from the transcript S. Then, Claim 13 follows from an application of the Chernoff bound to Inequality (42). We refer the reader to the proof of Claim 21 for details.

This finishes the proof of Claim 13. □

Next, we show the following Claim 14, which is the counterpart of Claim 22.

Claim 14 (Matching by Luck). *Let $Z'_{i,k}$ and Θ'_i be as defined in Algo. B.2. It holds that*

$$\Pr\left[\exists i \in \Sigma' \text{ s.t. } \textstyle\sum_{k\in[\ell_{\mathsf{gad}}]\setminus\Theta'_i} Z'_{i,k} < 60Q^7\lambda - 2Q^4\lambda\right] \le \mathsf{negl}(\lambda),$$

where the probability is taken over the whole random procedure Pick *as defined by Algo. 5.2.*

Proof. This proof is identical to that of Claim 22. We omit the details. □

Finishing the Proof of Claim 10. Similar as in the proof of Claim 5, for any session $i \in \mathsf{Corrupt}_1$ (resp. $i \in \mathsf{Corrupt}_2$), Claim 13 lower-bounds the number of matching Stage 2 (resp. Stage 1) gadgets in session i due to Watrous rewinding, and Claim 14 lower-bounds the number of matching Stage 2 (resp. Stage 2) gadgets in session i by pure luck.

Claims 13 and 14 together imply that the trapdoor witness must be available when the simulator needs it (except for with negligible probability). This step is identical to the argument presented in Appx. B.4. We omit the details.

6 Bounded-Concurrent PQ-2PCC and 2PQC

6.1 Construction

Our constructions of bounded-concurrent PQ-2PCC and 2PQC will follow the same template. In the following, we take PQ-2PCC as an example to explain the intuition.

Intuition. We need a constant-round PQ-2PCC protocol Π_{2PCC} in the CRS model, which has a straight-line simulator. That is, if P^*_b ($b \in \{0,1\}$) denote the corrupted party and x_b denotes its initial input, then the simulator $\mathcal{S}$ can be partitioned into two stages $\mathcal{S}_1$ and $\mathcal{S}_2$ that work in the following way:

- First, $\mathcal{S}_1(1^\lambda)$ outputs a fake CRS crs' together with a "trapdoor" td which is supposed to contain some information that helps $\mathcal{S}_2$ finish the simulation in straight-line. Importantly, $\mathcal{S}_1$ does not need to access the corrupted party.
- Then, $\mathcal{S}_2^{P^*_b}(\mathsf{crs}', \mathsf{td}, x_b)$ will finish the remaining job of simulation. Importantly, $\mathcal{S}_2$ needs access to the corrupted party P^*_b, but it does not rewind P^*_b. That is, it can generated the simulated execution by talking with P^*_b in straight-line.

If we have such a protocol Π_{2PCC} for an ideal $\mathcal{F}$, our construction will then proceed in two steps:

1. First, P_1 and P_2 run the bounded-concurrent coin-flipping protocol we constructed in Sec. 5 (i.e., Prot. 5) to securely generate a CRS crs;
2. Then, P_1 and P_2 execute the above constant-round Π_{2PCC} to compute $\mathcal{F}$. Note that Π_{2PCC} is in the CRS model; These two parties just use the crs generated in Step 1 as the required CRS.

Roughly speaking, this construction is secure (in the Q-concurrent setting) because the CRS generation step (which is implemented by Prot. 5) enjoys Q-concurrent security. But the actual security proof is more involved: Since Prot. 5 is only Q-concurrently secure (instead of being UC secure), we cannot claim directly that the above construction is also Q-concurrently secure, due to the existence of the extra constant-round Π_{2PCC} messages in each session. However, thanks to the straight-line behavior of the simulator for Π_{2PCC}, it can be easily composed with the simulator for Prot. 5, and eventually allows us to prove the Q-concurrent security of the above construction using a similar proof as for Prot. 5.

To do that, let us recall (from Sec. 5.2) how we prove the security of Prot. 5. We built a sequence of hybrids $\{G_{j,1}, \ldots, G_{j,6}\}_{j\in[L]}$, where

- $\{G_{j,1}, \ldots, G_{j,5}\}$ are for the following purpose: In a hybrid index by j, the Watrous rewinding is performed for the first $(j-1)$ blocks. The hybrid monitors the next block B_j. If an honest party's SBCom to his share in some session (say, session i) starts in block B_j, we ask this honest party in session i to use the $\mathsf{trapdoor}$ (extracted from the corresponding **Com-and-Guess Stage**) to cheat in its SBCom to his share, SFE, and WI. Our special design of **Com-and-Guess Stage** guarantees that the $\mathsf{trapdoor}$ is available when the (simulated) honest party of session i needs it, but the corrupted party cannot learn any $\mathsf{trapdoor}$. Note that in hybrid $G_{j,5}$, all the honest-party messages in the session i are already replaced by the simulated ones. That is, the simulation for this session i is essentially done at hybrid $G_{j,5}$.

– Then, $G_{j,6}$ goes to perform Watrous rewinding for the next block B_j so that the next iteration of hybrids $\{G_{j+1,1}, \ldots, G_{j+1,5}\}$ start to monitor the messages in block B_{j+1}, and finish the simulation for the sessions where the honest party's SBCom to his share starts in block B_{j+1}.

Now, to prove the security our current construction, we simply add one more hybrid $G_{j,5.5}$ between $G_{j,5}$ and $G_{j,6}$. As explained above, $G_{j,5}$ already finished the simulation for the underlying CRS generating protocol (i.e., the Step 1) for session i. That means hybrid $G_{j,5}$ already "forces" the corrupted party in session i to accept the simulated CRS crs' (generated by $\mathcal{S}_1(1^\lambda)$), which has a trapdoor td hidden from the corrupted party. Then, in hybrid $G_{j,5.5}$, we can use the Π_{2PCC} simulator $\mathcal{S}_2(\mathsf{crs}', \mathsf{td})$ to simulate the extra constant rounds of Π_{2PCC} (shown in Step 2). We can prove the indistinguishability and $\mathsf{invariant\ condition}$ for $G_{j,5.5}$ using exactly the same technique as for $\{G_{j,5}\}$—Since the $\mathcal{S}_2(\mathsf{crs}', \mathsf{td}')$ talks to the corrupted party in session i in straight-line, the simulation of these extra constant rounds is no difference from the changes for WI (or the SBCom and SFE) in earlier hybrids; Therefore, the indistinguishability and $\mathsf{invariant\ condition}$ can be proven using the same technique. Now, the new sequence of hybrids $\{G_{j,1}, \ldots, G_{j,5}, G_{j,5.5}, G_{j,6}\}_{j \in [L]}$ will allow us to finish the security proof.

Extension to 2PQC. Notice that the above security proof for the PQ-2PCC protocol makes use of the Π_{2PCC} in a "black-box" manner. That is, all the claims above hold as long as the Π_{2PCC} is constant-round and straight-line simulatable in the CRS model. In particular, it does *not* rely on the fact that the Π_{2PCC} messages are classical.

Therefore, a bounded-concurrent 2PQC can be constructed by replacing the Π_{2PCC} in the above protocol with a constant-round, straight-line simulatable (in the CRS model) 2PC for *quantum functionalities.*

Formal Construction. As discussed above, our constructions of the bounded-concurrent PQ-2PCC and 2PQC are almost identical, they only differ at the underlying stand-alone secure protocol. Therefore, we present them *at one stroke* in Prot. 6. This protocol makes use of all the building blocks as for our coin-flipping protocol shown in Sec. 5, i.e., SBCom, ENMC, WI, and SFE. Additional, we need a constant-round 2PQC protocol Π_{2PQC} (resp. a PQ-2PCC protocol Π_{2PCC}) in the CRS model that has a straight-line simulator. Such a Π_{2PQC} is known from the quantum hardness of LWE [BCKM21a]; Such a Π_{2PCC} is also known from the quantum hardness of LWE (e.g., [PVW08, GS18]).

Protocol 6: Bounded-Concurrent 2PQC (and PQ-2PCC)

Let $\mathcal{F}$ denote a quantum (resp. classical) functionality that two parties want to jointly evaluate. Let $Q(\lambda)$ be a polynomial of λ, denoting the maximum number of concurrent sessions. One party P_1 takes as input λ and a (potentially quantum) input $\mathbf{x}_1$; The other party P_2 takes as input λ and a (potentially quantum) input $\mathbf{x}_2$. P_1 and P_2 agree on an $\mathsf{id} \in \{0,1\}^\lambda$ as the ID for this execution of the protocol.

– **Stages 1-10:** P_1 and P_2 execute an instance of the bounded-concurrent tow-party coin-flipping protocol shown in Prot. 5. At the end of this phase, both P_1 and P_2 learn the same output crs.

  Comment. *We denote this phase as* **Stages 1-10** *because it is an execution of Prot. 5 which has ten stages.*

– **Stage 11:** P_1 and P_2 execute the constant-round 2PQC protocol Π_{2PQC} (resp. PQ-2PCC protocol Π_{2PCC} for the classical functionality) to compute the functionality $\mathcal{F}$. Note that this protocol is in the CRS model; P_1 and P_2 use the crs obtained from the previous phase as the CRS.

Security Proof. The security of Prot. 6 is established by the following Thm. 15, whose proof is provided in Sec. 6.2.

Theorem 15. *If Π_{2PQC} (resp. Π_{2PCC}) is a constant-round, straight-line simulatable 2PQC (resp. PQ-2PCC) protocol in the CRS model for a quantum (resp. classical) functionality $\mathcal{F}$, then Prot. 6 is a Q-concurrent 2PQC (resp. PQ-2PCC) for the same functionality $\mathcal{F}$.*

6.2 Security Proof (Proving Thm. 15)

As explain in the **Intuition** part in Sec. 6.1, the proof of Thm. 15 is almost identical to that of Thm. 9. Thus, in the following, we will only focus on the places where this proof differs from the one shown in Sec. 5.2.

Similar as in Sec. 5.2, we first recast the Q-concurrent execution of Prot. 6 into the MIM setting (similar to the one shown in Fig. 3), using exactly the same recasting strategy defined in the **Recast the Execution to the MIM Setting** part on Page 37.

Then, we partition this recast MIM execution into $L := 24Q^6\lambda$ equal-size blocks $\{B_1, \ldots, B_L\}$, in exactly the same manner as explained in the **Intuition** part on Page 39.

Next, we define a sequence of hybrids. We define hybrids REAL, $G_{0,6}$, and $\{G_{j,1}, \ldots, G_{j,6}\}_{j\in[L]}$ in the same way as in Sec. 5.2, except for the following slight modification of $\{G_{j,5}\}_{j\in[L]}$:

– Recall that in $\{G_{j,5}\}_{j\in[L]}$ (defined on Page 43), the hybrid makes use of the extracted share from the corrupted party to "force" the coin-flipping result (which is the crs in our current construction) to the one from the ideal functionality $\mathcal{F}_{\text{CP}}$. In our current proof, there is no $\mathcal{F}_{\text{CP}}$ any more. This hybrid simply run the simulator of the underlying Π_{2PQC} (resp. Π_{2PCC}) to learn the crs' to which the coin-flipping result should be enforced.
In more detail, we modify $\{G_{j,5}\}_{j\in[L]}$ as follows: It executes the straight-line simulator $(\mathsf{crs}', \mathsf{td}) \leftarrow \mathcal{S}_1(1^\lambda)$; Then, it stores td (for the use later in hybrid $G_{j,5.5}$ we will define soon) and set the (simulated) honest party's share to $r^{(i)}_b := \mathsf{crs}' \oplus r^{(i)}_{1-b}$, where $r^{(i)}_{1-b}$ is the corrupted party's share extracted from hybrid $G_{j,4}$ (b denotes the uncorrupted party). In this way, both parties will output crs' as the coin-flipping result. Other steps are performed in exactly the same manner as the $\{G_{j,5}\}_{j\in[L]}$ defined on Page 43. It is straightforward to see that this modification does not affect the indistinguishability and $\mathsf{invariant\ condition}$ of $\{G_{j,5}\}_{j\in[L]}$.

Next, we insert one more hybrid $\{G_{j,5.5}\}_{j\in[L]}$ between hybrid $\{G_{j,5}\}_{j\in[L]}$ and $\{G_{j,6}\}_{j\in[L]}$, which is defined as follows:

Hybrid $G_{j,5.5}$ ($\forall j \in [L]$): This hybrid is identical to $G_{j,5}$ except that $G_{j,5.5}$ uses the straight-line simulator of Π_{2PQC} (resp. Π_{2PCC}) to finish **Stage 11** of Prot. 6. In more detail:

– For any session i where $P^{(i)}_1$ is corrupted, if its Stage 4 SBCom starts in block B_j, then $P^{(i)}_2$ uses the straight-line simulator $\mathcal{S}_2^{P^{(i)}_1}(\mathsf{crs}', \mathsf{td}', \mathbf{x}^{(i)}_1)$ to talk with the corrupted $P^{(i)}_1$, where the crs' and td' are generated by $\mathcal{S}_1(1^\lambda)$ in $G_{j,5}$, and $\mathbf{x}^{(i)}_1$ is the corrupted $P^{(i)}_1$'s input.

– For any session i where $P^{(i)}_2$ is corrupted, if its Stage 3 SBCom starts in block B_j, then $P^{(i)}_1$ uses the straight-line simulator $\mathcal{S}_2^{P^{(i)}_2}(\mathsf{crs}', \mathsf{td}', \mathbf{x}^{(i)}_2)$ to talk with the corrupted $P^{(i)}_2$, where the crs' and td' are generated by $\mathcal{S}_1(1^\lambda)$ in $G_{j,5}$, and $\mathbf{x}^{(i)}_2$ is the corrupted $P^{(i)}_2$'s input.

This finishes the description of $G_{j,5.5}$.

As explained earlier, the straight-line simulator $\mathcal{S}_2$ for Π_{2PQC} (resp. Π_{2PCC}) works in straight-line and generates the simulated **Stage-11** messages that are computationally indistinguishable from a real execution of Π_{2PQC} (resp. Π_{2PCC}). Thus, the switch we made in hybrid $G_{k,5.5}$ is no different from our switch of witness in the WI (as we did in hybrid $G_{j,1}$ defined on Sec. 5.2). Therefore, we have the following lemma.

Lemma 20. *If Π_{2PQC} (resp. Π_{2PCC}) is constant-round and straight-line simulatable in the CRS model, then for any $j \in [L]$, it holds that $\mathsf{OUT}(G_{j,5}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{j,5.5})$. Moreover, if SBCom is computationally hiding, ENMC is both extractable (as per Def. 7) and first-message binding (as per Rmk. 4), and that the $\mathsf{invariant\ condition}$ holds in $G_{j,5}$ ($\forall j \in [L]$), then the $\mathsf{invariant\ condition}$ hold in $G_{j,5.5}$ ($\forall j \in [L]$) as well.*

Proof. The proof of this lemma is almost identical to that of Lem. 14. The only difference is: when proving $\mathsf{OUT}(G_{j,5}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{j,5.5})$, we rely on the security (i.e., straight-line simulatability) of Π_{2PQC} (resp. Π_{2PCC}), instead of the WI property of WI as in the proof of Lem. 14. We omit the details. □

Then, following exactly the same proof as for Lem. 19, we can show that $\mathsf{OUT}(G_{j,5.5}) \overset{\text{c}}{\approx} \mathsf{OUT}(G_{j,6})$ and the $\mathsf{invariant\ condition}$ in $G_{j,6}$.

Finally, by repeating the same argument as shown in the **Finishing the Proof** part on Page 44, we finish the proof of Thm. 15.

7 Bounded-Concurrent Multi-Party Coin-Flipping, PQ-MPCC, and MPQC

Our constructions in this section follow the same framework as for the two-party results shown earlier. We will first build a bounded-concurrent multi-party coin-flipping protocol (in Sec. 7.1). Then, bounded-concurrent *general-purpose* MPC for both classical and quantum functionality can be built from this multi-party coin-flipping protocol. The compiler is also similar to that from Sec. 6. That is, we first use the coin-flipping protocol to generate a CRS, and then run a constant-round, straight-line simulatable *general-purpose* MPC protocol in the CRS model using the generated CRS. Again, this step does not follow from a modular composition lemma (since the coin-flipping protocol is not *universally composable*); But we can still make the proof go through by performing "non-black-box" modifications to the proof for the coin-flipping protocol, in an almost identical manner as we build general-purpose 2PC from the two-party coin-flipping protocol (i.e., using the same techniques shown in Sec. 6.2).

7.1 Bounded-Concurrent Multi-Party Coin-Flipping: Construction

Our construction of multi-party coin-flipping protocol is a direct generalization of the two-party protocol shown in Sec. 5. The main difference is, we set the parameter ℓ_{gad} in a more generous manner to tolerate more concurrent instances of the **Com-and-Guess Stage** due to the increased number of parties. In particular, this ℓ_{gad} is set to $120\widehat{Q}^7\lambda$ with $\widehat{Q} \coloneqq n^2Q$. We refer to Sec. 7.2 for why this parameter setting suffices in the security proof.

Construction. Our construction makes use of the same building blocks as for our two-party coin-flipping protocol in Sec. 5, i.e., the SBCom, ENMC, SFE, and WI. The construction is presented in Prot. 7.

Protocol 7: Bounded-Concurrent Post-Quantum Multi-Party Coin-Flipping Protocol

Let $n(\lambda)$ be a polynomial of λ, denoting the number of parties in a session. Let $Q(\lambda)$ be a polynomial of λ, denoting the maximum number of concurrent sessions. All the parties take the security parameter λ as the common input. All pairs of two parties P_u and P_v are associated with a unique ID $\mathsf{id}_{u,v} \in \{0,1\}^\lambda$ for the communication between them.

1. **(Com-and-Guess.)** All pairs of parties P_u and P_v perform the following task.
   For $k = 1$ to $\ell_{\mathsf{gad}} \coloneqq 120\widehat{Q}^7\lambda$ where $\widehat{Q} \coloneqq n^2Q$, do the following *sequentially*:

   (a) P_v samples $b_{u,k} \xleftarrow{\$} \{0,1\}$ and commits to it using SBCom.

   (b) P_u samples $a_{u,k} \xleftarrow{\$} \{0,1\}$ and commits to it using ENMC. P_u and P_v will use $\mathsf{id}_{u,v}{:}u{:}k$ as the ID for this ENMC so that each ENMC is executed using a different ID.

   (c) P_v sends $b_{u,k}$ together with the decommitment information w.r.t. the SBCom in Step 1a. P_u continues only if this decommitment is valid.

   We remark that though the above repetition between P_u and P_v must be executed sequentially, this **Com-and-Guess Stage** between different pairs of parties can happen *in parallel*. For example, while P_1 and P_2 are performing this **Com-and-Guess Stage**, P_2 can perform the **Com-and-Guess Stage** with all the other $(n-2)$ parties (i.e., $P_3, \ldots, P_n$) simultaneously in parallel. So, this step is considered as $\ell_{\mathsf{gad}} \cdot (\mathit{\Gamma}_{\mathrm{SBC}} + \mathit{\Gamma}_{\mathrm{ENMC}})$ rounds. I.e., it does not *explicitly* depend on n; Rather, it depends on n only through ℓ_{gad}.

   Comment. *Again, we call each repetition a Stage 1* gadget*. The k-th Stage 1* gadget matches *if the $a_{u,k}$ committed by P_u in is equal to the $b_{u,k}$ validly* decommitted *by P_v. This step is designed to allow the simulator to learn a* trapdoor *witness (i.e., there are more than* $\mathsf{Th} \coloneqq 60Q^7\lambda + Q^4\lambda$ *Stage 1* gadgets *matching) when P_v is corrupted, while ensuring that the corrupted P_v cannot learn the* trapdoor *witness. We say that P_u is the* sender *and P_v is the* receiver *of this* **Com-and-Guess Stage**.

2. **(Commitment to P_u's Share.)** Each party P_u samples a random string r_u. Each pair of parties P_u and P_v execute a statistically-binding commitment SBCom where P_u commits to r_u.
   We remark that all the parties can perform this stage in parallel. So, this step is considered as $\mathit{\Gamma}_{\mathrm{SBC.C}}$ rounds (i.e., independent of the number of parties).

3. **(SFE with P_u as the Receiver.)** All pairs of parties P_u and P_v perform the following task. We first define a classical circuit C_u:

   – **Circuit C_u:** It hard-wires the value r_v and the transcript of Stage 1 **Com-and-Guess Stage** *where P_u is the* sender *and P_v is the* receiver; It takes as input a string w. If w is a witness for the fact that there are more than $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ Stage 1 gadgets matching in the hard-wired **Com-and-Guess Stage** transcript, it outputs r_u; Otherwise, it outputs a dummy symbol $\vdash$.

   P_u computes $\mathsf{k}_u \leftarrow \mathsf{SFE.Gen}(1^\lambda)$ and $\mathsf{ct}_u \leftarrow \mathsf{SFE.Enc}(\mathsf{k}_u, w_u)$, where w_u is set to an all-0 string. P_u sends ct_u to P_v. P_v computes $\widehat{\mathsf{ct}}_u \leftarrow \mathsf{SFE.Eval}(C_u, \mathsf{ct}_u)$ and sends $\widehat{\mathsf{ct}}_u$ to P_u. P_u simply ignores the message $\widehat{\mathsf{ct}}_u$. We remark that all the parties can perform this stage in parallel. So, this step is considered as $\mathit{\Gamma}_{\text{SFE}}$ rounds (i.e., independent of the number of parties).

   Comment. *This step is designed to allow the simulator learn P_v's committed share r_v, when P_v is corrupted. In that case, the simulator will set w_u to the* trapdoor *witness it learned from the* **Com-and-Guess Stage**, *so that she can learn r_v by decrypting $\widehat{\mathsf{ct}}_u$. Note that an honest P_u will always get $\vdash$ when decrypting $\widehat{\mathsf{ct}}_u$ (as w_u is set to an all-0 string). So the message $\widehat{\mathsf{ct}}_u$ is useless for an honest P_u.*

4. **(P_u Reveals r_u.)** Each P_u broadcasts the r_u that is committed in Stage 2. We emphasize that P_u sends the value r_u only, *without the associated decommitment information.*
   We remark that all the parties announce their own r_u simultaneously using the broadcast channel. So, this step is considered as a single round (i.e., independent of the number of parties).

5. **(P_u Proves Honesty.)** All pairs of parties P_u and P_v execute an instance of WI where P_u proves that

   (a) *either* its messages sent to P_v in Stages 2 to 4 are generated by honestly following the protocol; *or*

   (b) there are more that $\mathsf{Th} := 60Q^7\lambda + Q^4\lambda$ Stage 1 gadgets matching in the Stage 1 **Com-and-Guess Stage** *where P_u is the* sender *and P_v is the* receiver.

   An honest P_u will use the witness for Item 5a to finish this WI.
   We remark that that all the parties can perform this stage in parallel. So, this step is considered as $\mathit{\Gamma}_{\text{WI}}$ rounds (i.e., independent of the number of parties).

   Comment. *This step is designed to protect against a corrupted P_v. In that case, the simulator will use the* trapdoor *witness (i.e., witness for Item 5b) to finish this* WI *so that it does not need to perform Stages 2 to 4 honestly with the corrupted P_v. Meanwhile, we will show that even a cheating P_v cannot make the* trapdoor *witness available for her, so she has to perform Stages 2 to 4 honestly with other uncorrupted parties.*

Output: All parties then output $r := \oplus_{u=1}^{n} r_u$ as the coin-flipping result.

Security Proof. The security of Prot. 7 is established by the following Thm. 16, whose proof is provided in Sec. 7.2.

Theorem 16. *For any $Q(\lambda)$ that is a polynomial of the security parameter λ, Prot. 7 is a Q-concurrent post-quantum multi-party coin-flipping protocol.*

7.2 Bounded-Concurrent Multi-Party Coin-Flipping: Security Proof

The security proof follows the same template as for the two-party case shown in Sec. 5.2. First, we recast the Q-concurrent execution of Prot. 7 into the main-in-the-middle setting. The purpose of this step is to present the schedule of messages in a similar pattern as the MIM execution of Q-concurrent simulation-sound ZK we discussed in Sec. 4, so that we can re-use the same proof technique as for the Q-concurrent simulation-sound of Prot. 4.

Recast to the MIM Setting. To do that, we follow the same strategy as in the two-party setting (i.e., at the beginning of Sec. 5.2). Recall the the major principle is to recast the schedule so that: (1) $\mathcal{A}$ controlling all the corrupted parties sitting in the middle; (2) all the **Com-and-Guess Stags** where the receiver is

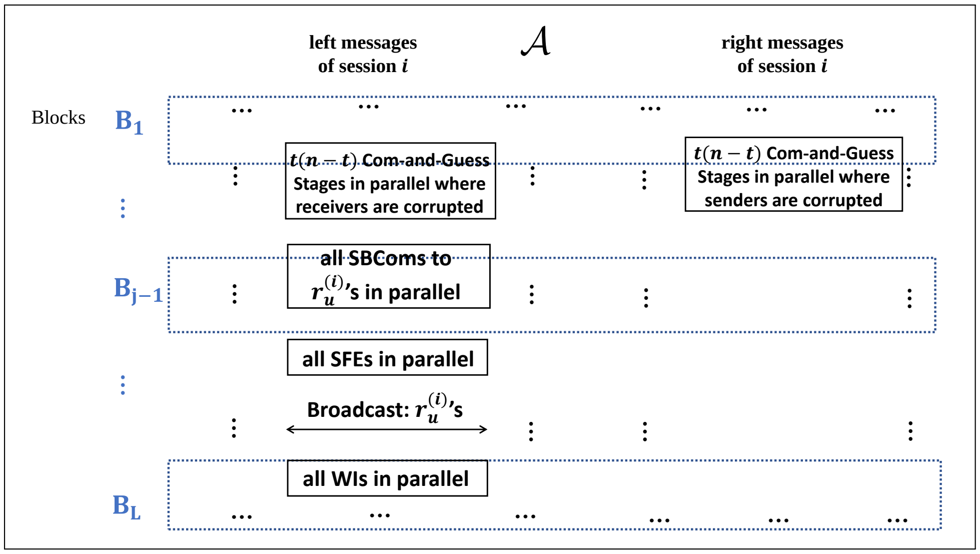

left messages of session i
$\mathcal{A}$
right messages of session i
Blocks
B_1
$t(n-t)$ Com-and-Guess Stages in parallel where receivers are corrupted
$t(n-t)$ Com-and-Guess Stages in parallel where senders are corrupted
B_{j-1}
all SBComs to $r_u^{(i)}$'s in parallel
all SFEs in parallel
Broadcast: $r_u^{(i)}$'s
all WIs in parallel
B_L

Fig. 4: Q-Concurrent Execution of Prot. 7 Recast to the MIM Setting. Only the messages exchanged between a corrupted party and an uncorrupted party are shown (In the shown session i, t parties are corrupted)

corrupted appear on the left; (3) all the **Com-and-Guess Stags** where the sender is corrupted appear on the right. As long as we recast the schedule in this way, the "simulation soundness" of the **Com-and-Guess Stags** will allow a simulator to learn a trapdoor for each **Com-and-Guess Stags** on the left, while ensuring that $\mathcal{A}$ that cannot learn any trapdoor from any **Com-and-Guess Stags** on the right.

The current multi-party case is a little more complex than the two-party case, because even in a fixed session, there are n parties and there are two **Com-and-Guess Stags** between each pair of parties with the reversed roles of sender and receiver. Thus, there are totally $2\cdot\binom{n}{2}$ **Com-and-Guess Stags** in a single session. The key observation here is that *we only need to focus on the* **Com-and-Guess Stags** *happen between a corrupted party and an uncorrupted party.* Assume that in session i, t parties are corrupted. Then, there will be exactly $t\cdot(n-t)$ **Com-and-Guess Stags** where the receiver is corrupted and $t\cdot(n-t)$ **Com-and-Guess Stags** where the sender is corrupted. We call these $2\cdot t\cdot(n-t)$ **Com-and-Guess Stags** "effective". The remaining $2\cdot\binom{n}{2}-2\cdot t\cdot(n-t)$ **Com-and-Guess Stags** must happen *either* between two uncorrupted parties *or* two corrupted parties, and we do not need to consider them when recasting the schedule. Then, we can recast the schedule into the pattern shown in Fig. 4, where only the effective **Com-and-Guess Stags** will appear on the left or right side.

Note that Fig. 4 is almost identical to (the recast) schedule of the Q-concurrent two-party setting (i.e., Fig. 3). The only difference is, in Fig. 3, there are exactly Q **Com-and-Guess Stags** on the the left (and right) side where each session contributes exactly one **Com-and-Guess Stag** on each side. However, in Fig. 4, each session will contribute $t\cdot(n-t)$ **Com-and-Guess Stag** on the left, and $t\cdot(n-t)$ **Com-and-Guess Stag** on the right. But our proof will still go through because we modified the parameters: In Prot. 7, we change ℓ_{gad} from $120Q^7\lambda$ to $120\widehat{Q}^7\lambda$ with $\widehat{Q}:=n^2Q$. This is exactly to reconcile the fact that each session in Fig. 3 will contribute more than one (but less than n^2, since $t\cdot(n-t)<n^2$) **Com-and-Guess Stag** on each side.

Again, there are other stages apart from the **Com-and-Guess Stages** in Prot. 7. They can be put on either side in Fig. 4 and it will not affect our proof (roughly, that is because they have only constant rounds for each session). Similar as in the two-party setting, we choose to put all these stages on the left.

We now formally describe the recast schedule. For all $i\in[Q]$, let $\mathsf{Corrupt}^{(i)}$ be a subset of $[n]$ that $P_u^{(i)}$ is corrupted iff $u\in\mathsf{Corrupt}^{(i)}$, and let $\mathsf{Honest}^{(i)}:=[n]\setminus\mathsf{Corrupt}^{(i)}$. We first determine the messages that appear on the right. The right interaction consists of the following messages:

- For all session $i\in[Q]$, the Stage 1 messages between $P_u^{(i)}$ and $P_v^{(i)}$, where $P_u^{(i)}$ with $u\in\mathsf{Corrupt}^{(i)}$ is the sender and $P_v^{(i)}$ with $v\in\mathsf{Honest}^{(i)}$ is the receiver.

All other stages will be put on the left. That is, the left interaction consists of the following messages

– For all session $i \in [Q]$, the Stage 1 messages between $P_u^{(i)}$ and $P_v^{(i)}$, where $P_u^{(i)}$ with $u \in \mathsf{Honest}^{(i)}$ is the sender and $P_v^{(i)}$ with $v \in \mathsf{Corrupt}^{(i)}$ is the $\mathsf{receiver}$.
– For all session $i \in [Q]$, all other stages (i.e., except for Stage 1) of session i.

This recast MIM schedule is depicted in Fig. 4.

Block Rewindings. With the recast schedule shown in Fig. 4, the remaining proof is identical to that for the two-party case (shown in Sec. 5.2). The only difference is that when we divide the messages into equal-size blocks, we only consider the effective messages (shown on the left and right sides in Sec. 5.2), and ignores the messages exchanged among uncorrupted parties (or among corrupted parties).

In more details, let T denote the total number of these effective messages. We will again partition these T messages (as shown in Fig. 3) into $L \coloneqq 24\widehat{Q}^6\lambda$ equal-size blocks $\{B_1, \ldots, B_L\}$. That is, block B_1 contains the first $\frac{T}{L}$ messages, block B_2 contains the next $\frac{T}{L}$ messages, and so on (messages are ordered according to their order of appearing in the execution, *skipping the messages that are not* effective).

Then, we define a similar sequence of hybrids as in Sec. 5.2, to perform the following block-rewinding simulation:

1. In the hybrids indexed by $j \in [L]$, we perform Watrous rewindings for the first $(j-1)$ blocks and monitor the execution of messages in block j. If there is at least one a fully nested left gadget[36] in block B_j, the hybrid pick one of these fully nested left gadget and rewind the whole block B_j iff this gadget does not match; Otherwise, a "dummy rewinding" is performed for this block with probability $\frac{1}{2}$.
This is the counterpart step of the $G_{0,6}$ in the two-party case (defined on Page 40).
2. If all the left **Com-and-Guess Stages** of some session i completes in block B_j (note that the broadcast SBComs to $r_u^{(i)}$'s in Fig. 4 signals the completion of all **Com-and-Guess Stages** of session i), it means the hybrid learned a trapdoor from each of these **Com-and-Guess Stages** of session i. This can be proven following the same argument as in the tow-party setting (i.e., Sec. 5.3) together with our new parameters $\ell_{\mathsf{gad}} = 120\widehat{Q}^7\lambda$ and $L = 24\widehat{Q}^6\lambda$ where $\widehat{Q} \coloneqq n^2Q$.
The hybrid then use the extracted trapdoor to "cheat" in his WI, SFE, and SBCom to r_u one-by-one, so that it effectively enforces the coin-flipping result to the one from the ideal functionality. Meanwhile, because of the invariant condition (which can be defined and proven using the same technique as in the two-party setting), $\mathcal{A}$ cannot cheat. These are the counterpart steps of the $G_{j,1}$ to $G_{j,5}$ in the two-party case (on Pages 41 and 43).
3. After the above changes, we go to perform Watrous rewinding for the j-th block (this is the counterpart step of the $G_{j,6}$ in the two-party case (on Page 44)), and then enter the next iteration of hybrids to deal with the $(j+1)$-th block.

Using the same arguments as in Sec. 5.2, we can prove the view indistinguishability and invariant condition in all these hybrids, which further implies the security of this multi-party coin-flipping protocol in the Q-concurrent setting. Since the same proof has been used earlier for our ZK arguments, two-party coin-flipping protocols, and PQ-2PCC and 2PQC, we believe the reader is already familiar with it. So, the details are omitted.

7.3 Bounded-Concurrent PQ-MPCC and MPQC

Given the bounded-concurrent multi-party coin-flipping protocol, we can build bounded-concurrent MPC for both classical and quantum functionality following the same template as in the two-party setting. That is, we take a constant-round, straight-line simulatable PQ-MPQC protocol Π_{MPQC} (or a MPCC protocol Π_{MPCC} for quantum functionalities) in the CRS model as the base protocol. We then compile this base protocol into a bounded-concurrent protocol in the following way: (1) all parties first run the multi-party coin-flipping protocol we built above to generate a CRS; (2) they then execute the base protocol (Π_{MPQC} or Π_{MPCC}) using the generated CRS. Such a Π_{MPQC} is known from the quantum hardness of LWE [BCKM21a]; Such a Π_{MPCC} is also known from the quantum hardness of LWE (e.g., [PVW08, GS18]).

[36] Here, "left gadget" means a gadget from some Com-and-Guess Stage appearing on the left side in Fig. 4.

We present the construction in Prot. 8. Similar as Prot. 6, we present the protocols for both classical and quantum functionalities at one stroke.

Protocol 8: Bounded-Concurrent MPQC (and PQ-MPCC)

Let $n(\lambda)$ be a polynomial of the security parameter λ. Let $\mathcal{F}$ denote a quantum (resp. classical) functionality that n parties want to jointly evaluate. Let $Q(\lambda)$ be a polynomial of λ, denoting the maximum number of concurrent sessions. Party P_u takes as input λ and a (potentially quantum) input $\mathbf{x}_u$. All pairs of two parties P_u and P_v are associated with an unique ID $\mathsf{id}_{u,v} \in \{0,1\}^\lambda$ for the communication between them.

– **Stages 1-5:** The n parties execute an instance of the bounded-concurrent multi-party coin-flipping protocol shown in Prot. 7. At the end of this phase, all parties learn the same output crs.

  Comment. *We denote this phase as* **Stages 1-5** *because it is an execution of Prot. 7 which has five stages.*

– **Stage 6:** The n parties execute the constant-round MPQC protocol Π_{MPQC} (resp. PQ-MPCC protocol Π_{MPCC} for the classical functionality) to compute the functionality $\mathcal{F}$. Note that this protocol is in the CRS model; The parties use the crs obtained above as the CRS.

To prove the security of Prot. 8 in the Q-concurrent setting, notice that this protocol is almost identical to our multi-party coin-flipping protocol (i.e., Prot. 7), except that there are extra constant rounds (i.e., the Π_{MPCC} or Π_{MPQC} messages) appended at the end of each session. These extra rounds can be treated as a new constant-round component whose security proof is in straight-line (assuming the trapdoor of the CRS is available to the simulator $\mathcal{S}_2$, which is true due to the security of the multi-party coin-flipping protocol executed to generate the CRS). Therefore, the security of Prot. 8 can be shown following exactly the same techniques, where we will deal with these extra Π_{MPCC} (or Π_{MPQC}) messages in the same manner as we deal with the WI arguments (or the commitment or SFE) in the security of our multi-party coin-flipping protocol in Sec. 7.1. Since the same proof has appeared several times before (for our ZK arguments, two-party coin-flipping protocols, and PQ-2PCC and 2PQC), we omit the details.

8 Impossibility of (Unbounded) Concurrent 2PQC

In this section, we show that there exists a functionality for which realizing an unbounded concurrent 2PQC is impossible, even with quantum communication. We start by recalling a few useful building blocks (in Sec. 8.1 to 8.3) that will be instrumental to prove our theorem (in Sec. 8.4 and 8.5).

8.1 Quantum Teleportation

We briefly recall the quantum teleportation [BBC+93] procedure, which is a protocol between Alice and Bob, sharing the maximally entangled state

$$\mathbf{e} = \frac{|00\rangle + |11\rangle}{\sqrt{2}},$$

where Alice wants to send a state $\boldsymbol{\rho}$ to Bob. Alice measures her registers in the

$$\left\{ \frac{|00\rangle + |11\rangle}{\sqrt{2}}, \frac{|00\rangle - |11\rangle}{\sqrt{2}}, \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \frac{|01\rangle - |10\rangle}{\sqrt{2}} \right\}$$

basis and returns the bits (x, z), denoting the outcome of the measurement, to Bob. Using this information, Bob can apply the map X^x, Z^z to his local state to recover $\boldsymbol{\rho}$. For the purpose of this work, it suffices to use the fact that, for all states $\boldsymbol{\rho}$ (for convenience, we only describe the experiment for single-qubit states), the following two experiments are identical, from the perspective of Bob. The first experiment proceeds as follows.

– Alice sends one qubit of the state $\mathbf{e}$ to Bob.

– Alice teleports $\boldsymbol{\rho}$ using the above procedure, and sends to Bob the output of the teleportation measurement.

Whereas the second experiment is defined below.

– Alice samples two random bits (x,z) and sends $X^x Z^z \boldsymbol{\rho}$ to Bob.

– Alice returns (x,z) to Bob.

The fact that these two experiments are identical follows from the fact that, from Bob's perspective, the state sent in the first step is maximally mixed in both experiments.

8.2 Simulation Secure Quantum Message Authentication

We recall the notion of quantum message authentication, and in particular the definition with simulation security from Broadbent and Wainewright [BW16]. In the same work, the authors show that the Clifford code and the trap code satisfy this strong notion of security.

Definition 17 (Quantum MAC). *A quantum message authentication code (MAC) consists of a polynomial size set of encoding and decoding channels*

$$\left\{\mathsf{Encode}^{(k)}_{\mathtt{M}\to\mathtt{C}}, \mathsf{Decode}^{(k)}_{\mathtt{C}\to\mathtt{MF}}, \right\}_{k\in K(\lambda)}$$

where $K(\lambda)$ is the set of all possible keys, $\mathtt{M}$ is the message register, $\mathtt{C}$ is the codeword register, and $\mathtt{F}$ is a flag register spanned by two orthogonal states $|\mathsf{acc}\rangle\langle\mathsf{acc}|$ and $|\mathsf{rej}\rangle\langle\mathsf{rej}|$. We require the following properties.

– ***Correctness:*** *For all $\lambda\in\mathbb{N}$, all $k\in K(\lambda)$, and all $\boldsymbol{\rho}_{\mathtt{M}}$ it holds that*

$$\left(\mathsf{Decode}^{(k)}_{\mathtt{C}\to\mathtt{MF}}\circ\mathsf{Encode}^{(k)}_{\mathtt{M}\to\mathtt{C}}\right)(\boldsymbol{\rho}_{\mathtt{M}}) = \boldsymbol{\rho}_{\mathtt{M}}\otimes|\mathsf{acc}\rangle\langle\mathsf{acc}|\,.$$

– ***Simulation Security:*** *For a fixed key k, let us define the real channel as*

$$\mathsf{Real}^{(k)}_{\mathtt{MR}\to\mathtt{MRF}}:\boldsymbol{\rho}_{\mathtt{MR}}\mapsto\left(\mathsf{Decode}^{(k)}_{\mathtt{C}\to\mathtt{MF}}\otimes\mathbb{I}_{\mathtt{R}}\right)\left(\mathcal{A}_{\mathtt{MR}}\left(\mathsf{Encode}^{(k)}_{\mathtt{M}\to\mathtt{C}}\otimes\mathbb{I}_{\mathtt{R}}\right)(\boldsymbol{\rho}_{\mathtt{MR}})\mathcal{A}^{\dagger}_{\mathtt{MR}}\right)$$

where $\mathcal{A}$ is an adversarial unitary and $\mathtt{R}$ is some auxiliary input register. Similarly, let us define the ideal channel as

$$\mathsf{Ideal}_{\mathtt{MR}\to\mathtt{MRF}}:\boldsymbol{\rho}_{\mathtt{MR}}\mapsto\left(\mathbb{I}_{\mathtt{M}}\otimes\mathcal{S}^{(\mathsf{acc})}_{\mathtt{R}}\right)\boldsymbol{\rho}_{\mathtt{MR}}\otimes|\mathsf{acc}\rangle\langle\mathsf{acc}| + tr_{\mathtt{M}}\left(\left(\mathbb{I}_{\mathtt{M}}\otimes\mathcal{S}^{(\mathsf{rej})}_{\mathtt{R}}\right)\boldsymbol{\rho}_{\mathtt{MR}}\right)\otimes\Omega\otimes|\mathsf{rej}\rangle\langle\mathsf{rej}|$$

where Ω is some fixed state and $\mathcal{S}^{(\mathsf{acc})}_{\mathtt{R}}$ and $\mathcal{S}^{(\mathsf{rej})}_{\mathtt{R}}$ are two completely-positive (CP) maps acting on $\mathtt{R}$ such that $\mathcal{S}^{(\mathsf{acc})}_{\mathtt{R}}+\mathcal{S}^{(\mathsf{rej})}_{\mathtt{R}}=\mathbb{I}$.
We say that a quantum MAC is simulation-secure if for all $\lambda\in\mathbb{N}$, all states $\boldsymbol{\rho}_{\mathtt{MR}}$, all polynomial-time unitaries $\mathcal{A}$, there exists a pair of polynomial-time CP maps $(\mathcal{S}^{(\mathsf{acc})}_{\mathtt{R}},\mathcal{S}^{(\mathsf{rej})}_{\mathtt{R}})$ such that the trace distance between the following states

$$\frac{1}{|K(\lambda)|}\sum_{k\in K(\lambda)}\mathsf{Real}^{(k)}_{\mathtt{MR}\to\mathtt{MRF}}(\boldsymbol{\rho}_{\mathtt{MR}})\approx\mathsf{Ideal}_{\mathtt{MR}\to\mathtt{MRF}}(\boldsymbol{\rho}_{\mathtt{MR}})$$

is negligible in λ.

8.3 Adaptively Secure Quantum Garbled Circuits

For the impossibility result, we need a notion of quantum garbled circuits that is adaptively secure. We show that such a quantum garbling scheme can be obtained by slightly modifying the construction from [BY22]. We first recall the notion of quantum garbled circuits [BY22]. As will become clear in our proof, we need the quantum garbling scheme to enjoy *decomposable* input encoding, i.e., each of the n input qubits is encoded individually. In the following Def. 18, we define explicitly the decomposable variant. The construction from [BY22], which only assumes the existence of post-quantum OWFs, satisfies Def. 18.

Definition 18 (Quantum Garbled Circuits). *A garbling scheme for Clifford plus measurement quantum circuits consists of three procedures* $(\mathsf{QGarble}, \mathsf{QGEncode}, \mathsf{QGEval}, \mathsf{QGSim})$ *with the following syntax.*

– $\mathsf{QGarble}(1^\lambda, Q, \mathbf{e}_{0,1}, \ldots, \mathbf{e}_{0,n}, r)$*: A QPT procedure that takes as input the security parameter* 1^λ*, a quantum circuit* Q*,* n *registers containing half of a maximally entangled state* $\mathbf{e}_{0,1}, \ldots, \mathbf{e}_{0,n}$*, and some local randomness* r*, and returns a quantum garbled circuits* $\widetilde{Q}$*, along with* n *classical strings* $r_1, \ldots, r_n$*.*

– $\mathsf{QGEncode}(i, r_i, \mathbf{e}_{1,i}, \mathbf{x}_i)$*: A QPT procedure that takes as input and index* $i \in [n]$*, a classical string* r_i*, a register containing half of a maxiamally enangled state* $\mathbf{e}_{1,i}$*, and a qubit* $\mathbf{x}_i$ *and returns an encoding* $\widetilde{E}_i$*.*

– $\mathsf{QGEval}(\widetilde{E}_1, \ldots, \widetilde{E}_n, \widetilde{Q})$*: A QPT procedure that takes as input a garbled input* $\widetilde{E}_1, \ldots, \widetilde{E}_n$ *and a garbled circuit* $\widetilde{Q}$ *and returns an output state* $\boldsymbol{\phi}$*.*

– $\mathsf{QGSim}(1^\lambda, \mathsf{par}, \boldsymbol{\phi})$*: A QPT procedure that takes as input the security parameter* 1^λ*, parameters* par *for a quantum circuit, and an output state* $\boldsymbol{\phi}$*, and outputs a simulated garbled input and garbled circuit* $(\widetilde{E}_1, \ldots, \widetilde{E}_n, \widetilde{Q})$*.*

We require the following properties.

– ***Correctness:*** *For all* $\lambda \in \mathbb{N}$*, all circuits* Q *with parameters* par*, all states* $\boldsymbol{\psi}$ *along with a (possibly entangled) auxiliary input* $\mathbf{z}$*, it holds that the following two states are statistically close:*

$$\left\{\mathbf{z}, \mathsf{QGEval}(\widetilde{E}_1, \ldots, \widetilde{E}_n, \widetilde{Q})\right\} \overset{s}{\approx} \{\mathbf{z}, Q(\boldsymbol{\psi})\}$$

where $(\widetilde{Q}, r_1, \ldots, r_n) \leftarrow \mathsf{QGarble}(1^\lambda, Q, \mathbf{e}_{0,1}, \ldots, \mathbf{e}_{0,n}, r)$*,* $\widetilde{E}_i \leftarrow \mathsf{QGEncode}(i, r_i, \mathbf{e}_{1,i}, \boldsymbol{\psi}_i)$ *for all* $i \in [n]$*, and* $\mathbf{e}_{0,i}$ *and* $\mathbf{e}_{1,i}$ *(for each* $i \in [n]$*) are the two halves of a maximally entangled state.*

– ***(Selective) Security:*** *For all* $\lambda \in \mathbb{N}$*, all circuits* Q *with parameters* par*, all states* $\boldsymbol{\psi}$ *along with a (possibly entangled) auxiliary input* $\mathbf{z}$*, it holds that the following two states are computationally close:*

$$\left\{\mathbf{z}, \widetilde{E}_1, \ldots, \widetilde{E}_n, \widetilde{Q}\right\} \overset{c}{\approx} \{\mathbf{z}, \mathsf{QGSim}(1^\lambda, \mathsf{par}, Q(\boldsymbol{\psi}))\}.$$

where $(\widetilde{Q}, r_1, \ldots, r_n) \leftarrow \mathsf{QGarble}(1^\lambda, Q, \mathbf{e}_{0,1}, \ldots, \mathbf{e}_{0,n}, r)$*,* $\widetilde{E}_i \leftarrow \mathsf{QGEncode}(i, r_i, \mathbf{e}_{1,i}, \boldsymbol{\psi}_i)$*, and* $\mathbf{e}_{0,i}$ *and* $\mathbf{e}_{1,i}$ *are the two halves of a maximally entangled state.*

Adaptive Security. In this work, we are going to require the quantum garbled circuit to satisfy the stronger notion of *adaptive* simulation security, where the distinguisher can choose the input state $\boldsymbol{\psi}$ adaptively, possibly depending on the garbled circuit $\widetilde{Q}$. In the following, we provide a generic transformation that turns any selectively secure garbling scheme into an adaptively secure one.

– **Garbling:** Sample n EPR pairs $\{(\mathbf{e}_{0,i}, \mathbf{e}_{1,i})\}_{i\in[n]}$. Run the selectively secure algorithm $(\widetilde{Q}, r_1, \ldots, r_n) \leftarrow \mathsf{QGarble}(1^\lambda, Q, \mathbf{e}_{0,1}, \ldots, \mathbf{e}_{0,n}, r)$. Store $(r_1, \ldots, r_n)$ for the use of the input encoding procedure, and return

$$\widetilde{R} = X^{\widetilde{x}} Z^{\widetilde{z}} \widetilde{Q}$$

where $\widetilde{x} \in \{0,1\}^m$ and $\widetilde{z} \in \{0,1\}^m$ are uniformly sampled bit-strings, for an m-qubit state $\widetilde{Q}$, which are also kept as part of the input encoding.

– **Input Encoding:** Compute $\widetilde{E}_i \leftarrow \mathsf{QGEncode}(i, r_i, \mathbf{e}_{1,i}, \mathbf{x}_i)$ for all $i \in [n]$ and, in addition, return $\widetilde{x}, \widetilde{z}$.

– **Evaluation:** Compute $\widetilde{Q} = X^{\widetilde{x}} Z^{\widetilde{z}} \widetilde{R}$ and run $\mathsf{QGEval}(\widetilde{Q}, \widetilde{E}_1, \ldots, \widetilde{E}_n)$.

– **Simulation:** Sample m EPR pairs $\{(\mathbf{r}_{0,i}, \mathbf{r}_{1,i})\}_{i\in[m]}$. . Set $\widetilde{R}$ to be a collection of m registers, each containing the qubit $\mathbf{r}_{0,i}$. Sent $\widetilde{R}$ to the distinguisher as the (simulated) garbled circuit. Upon receiving the output state $\boldsymbol{\phi}$, run the simulator of the selectively secure scheme

$$(\widetilde{E}_1, \ldots, \widetilde{E}_n, \widetilde{Q}) \leftarrow \mathsf{QGSim}(1^\lambda, \mathsf{par}, \boldsymbol{\phi})$$

For each qubit of $\widetilde{Q}$, apply a teleportation measurement together with $\mathbf{r}_{1,i}$, and denote by x_i, z_i the output bits of the measurement. Return $\widetilde{x} = (x_1, \ldots, x_m)$, $\widetilde{z} = (z_1, \ldots, z_m)$, along with $\widetilde{E}_1, \ldots, \widetilde{E}_n$.

It is easy to see that the scheme is correct. To see why the scheme satisfies adaptive security, it suffices to observe that, from the point of view of the distinguisher, the state $\widetilde{R}$ is the maximally mixed state in both the real and the simulated distribution. Therefore, the distribution induced by the teleportation measurement is identical to the real one (see also Sec. 8.1). Indistinguishability then follows from the indistinguishability of the selectively secure garbling scheme.

8.4 Oracle Separation

Our impossibility result will follow closely the outline of [BPS06], although with some non-trivial modification to handle the quantum nature of the protocol. We will start with the following warm-up result in an oracular model, which will be a useful intermediate step for our final result.

Lemma 21. *Let f be a one-way function, and let R_f be the following NP-relation*

$$R_f = \{(x, w) : f(w) = x\}.$$

Let Π^{ZK} be a stand-alone zero-knowledge proof of knowledge for R_f with $\ell = \ell(\lambda)$ prover messages. There exists a functionality $\mathcal{G}$, a distribution $\mathcal{D}$, a function secret, *and a polynomial-time adversary $\mathcal{A}$ such that:*

- *In a concurrent execution scheduled by $\mathcal{A}$ of one copy of Π^{ZK} (with $\mathcal{A}$ playing the role of the verifier) and ℓ ideals calls to $\mathcal{G}$ with $\mathcal{A}$ providing the second input and receiving the output, if the inputs to the honest parties are chosen from $d \leftarrow \mathcal{D}$, then $\mathcal{A}$ learns* $\mathsf{secret}(d)$ *with probability one.*
- *In any execution of ℓ copies of the ideal calls to $\mathcal{G}$ and a copy of the ideal functionality $\mathcal{F}^{\mathsf{ZK}}$, with honest inputs chosen from $d \leftarrow \mathcal{D}$, any polynomial time adversary $\widetilde{\mathcal{A}}$ will only output* $\mathsf{secret}(d)$ *with negligible probability.*

Proof. Let $\mathsf{Ver}^{\mathsf{ZK}}_{\mathtt{SZJ}\to\mathtt{SZ}}$ be the channel that computes the next message function of the verifier, where $\mathtt{S}$ is the register storing the internal state of the verifier, $\mathtt{Z}$ is the register storing the message of the ZK protocol, and $\mathtt{J}$ is the register storing the round counter. We define the two-party functionality $\mathcal{G}$ as follows.

- Sender Input: The keys $(k_1 \dots, k_{\ell-1}) \in K(\lambda)$, the seeds $(r_2, \dots, r_\ell) \in \{0,1\}^\lambda$, a secret $s \in \{0,1\}^\lambda$, and a basis state $\boldsymbol{\rho}_{\mathtt{S}}$.
- Receiver Input: A state $\boldsymbol{\rho}_{\mathtt{ZC}}$, a round counter j, and a seed $\widetilde{r_j}$.
- The functionality computes its output as follows.
  - If $j = 1$: Apply the map

$$\left(\left(\mathsf{Encode}^{(k_1)}_{\mathtt{S}\to\mathtt{C}} \otimes \mathbb{I}_{\mathtt{Z}}\right) \circ \mathsf{Ver}^{\mathsf{ZK}}_{\mathtt{SZJ}\to\mathtt{SZ}}\right) (\boldsymbol{\rho}_{\mathtt{S}} \otimes tr_{\mathtt{C}}(\boldsymbol{\rho}_{\mathtt{ZC}}) \otimes |1\rangle\langle 1|) \mapsto \boldsymbol{\rho}_{\mathtt{CZ}}$$

    and return the resulting state $\boldsymbol{\rho}_{\mathtt{CZ}}$ to the adversary, along with r_2.
  - If $1 < j < \ell$: Check if $\widetilde{r_j} = r_j$ and return $\perp$ if this is not the case. Apply the map

$$\left(\mathsf{Decode}^{(k_{j-1})}_{\mathtt{C}\to\mathtt{SF}} \otimes \mathbb{I}_{\mathtt{Z}}\right) (\boldsymbol{\rho}_{\mathtt{ZC}}) \mapsto \boldsymbol{\rho}_{\mathtt{SZF}}$$

    and project the $\mathtt{F}$ register onto $|\mathsf{acc}\rangle\langle\mathsf{acc}|$. Return $\perp$ if the projection fails. Else denote by $\boldsymbol{\rho}_{\mathtt{SZ}}$ the partial trace on registers $\mathtt{S}$ and $\mathtt{Z}$ and compute the mapping

$$\left(\left(\mathsf{Encode}^{(k_j)}_{\mathtt{S}\to\mathtt{C}} \otimes \mathbb{I}_{\mathtt{Z}}\right) \circ \mathsf{Ver}^{\mathsf{ZK}}_{\mathtt{SIJ}\to\mathtt{SZ}}\right) (\boldsymbol{\rho}_{\mathtt{SZ}} \otimes |j\rangle\langle j|) \mapsto \boldsymbol{\rho}_{\mathtt{CZ}}.$$

    Return the resulting state $\boldsymbol{\rho}_{\mathtt{CZ}}$ to the adversary, along with r_{j+1}.
  - If $j = \ell$: Check if $\widetilde{r_\ell} = r_\ell$ and return $\perp$ if this is not the case. Apply the map

$$\left(\mathsf{Decode}^{(k_{\ell-1})}_{\mathtt{C}\to\mathtt{SF}} \otimes \mathbb{I}_{\mathtt{Z}}\right) (\boldsymbol{\rho}_{\mathtt{ZC}}) \mapsto \boldsymbol{\rho}_{\mathtt{SZF}}$$

and project the $\mathtt{F}$ register onto $|\mathsf{acc}\rangle\langle\mathsf{acc}|$. Return $\perp$ if the projection fails. Else denote by $\boldsymbol{\rho}_{\mathtt{SZ}}$ the partial trace on registers $\mathtt{S}$ and $\mathtt{Z}$ and compute the mapping

$$\mathsf{Ver}^{\mathsf{ZK}}_{\mathtt{SZJ}\to\mathtt{SZ}}\left(\boldsymbol{\rho}_{\mathtt{SZ}} \otimes |\ell\rangle\langle\ell|\right) \mapsto \boldsymbol{\rho}_{\mathtt{SZ}}.$$

Check if the output state corresponds to an accepting state of the verifier and return s if this is the case, otherwise return $\perp$.

First, we argue that an attacker with a concurrent scheduling of Π^{ZK} can recover the secret s with probability 1. This is because the attacker can simply forward the messages of the prover to the verifier, along with the register $\mathtt{C}$ while keeping it untouched. Since the (honest) prover is assumed to succeed with probability 1, then so does the attacker.

Second, we need to show that any attacker $\widetilde{\mathcal{A}}$ interacting with the ideal functionality, cannot extract the secret s, except with negligible probability. Our first step is to establish that the ℓ queries of the adversary must correspond to a straight-line execution, i.e., they must be on input the counter $1, \dots, \ell$ in increasing order. Consider the ℓ-th query: We claim that, with all but negligible probability, it must be the case that either the output is $\perp$ (in which case the adversary does not learn s) or that the adversary must have previously queried the oracle on counter $\ell - 1$. This is because the adversary needs to guess the value of r_ℓ to receive a non-$\perp$ output, which is only sent to the attacker on a query on input $\ell - 1$. Applying this argument recursively, and using the fact that the adversary has *at most* ℓ queries at their disposal, we obtain the desired implication. The same argument also shows that all queries of the adversary must be non-$\perp$.

Next, we show that $\widetilde{\mathcal{A}}$ must be computing a valid zero-knowledge proof. Without loss of generality we model the adversary's next message function as a unitary U acting on some internal register $\mathtt{A}$, on the message register $\mathtt{Z}$, and on the codeword register $\mathtt{C}$. Let $\boldsymbol{\rho}_{\mathtt{SZA}}$ be the state consisting of the output by one application of the $\mathsf{Ver}^{\mathsf{ZK}}_{\mathtt{SZJ}\to\mathtt{SZ}}$ channel during the j-th round, along with the current internal state of the attacker. Then, by construction, the state undergoes the following evolution before being fed again into the verifier channel

$$\left(\mathsf{Decode}^{(k_j)}_{\mathtt{C}\to\mathtt{SF}} \otimes \mathbb{I}_{\mathtt{ZA}}\right)\left(U\left(\mathsf{Encode}^{(k_j)}_{\mathtt{S}\to\mathtt{C}} \otimes \mathbb{I}_{\mathtt{ZA}}\right)(\boldsymbol{\rho}_{\mathtt{SZA}})U^\dagger\right).$$

By Def. 17 this state is negligibly close in trace distance to the state produced by the simulator acting only on registers $\mathtt{Z} \otimes \mathtt{A}$. Since we are post-selecting on the decoding channel being accepting, we obtain that

$$tr_{\mathtt{F}}\left(\left(\mathsf{Decode}^{(k_j)}_{\mathtt{C}\to\mathtt{SF}} \otimes \mathbb{I}_{\mathtt{ZA}}\right)\left(U\left(\mathsf{Encode}^{(k_j)}_{\mathtt{S}\to\mathtt{C}} \otimes \mathbb{I}_{\mathtt{ZA}}\right)(\boldsymbol{\rho}_{\mathtt{SZA}})U^\dagger\right)\right) \approx \left(\mathbb{I}_{\mathtt{S}} \otimes \mathcal{S}^{(\mathsf{acc})}_{\mathtt{ZA}}\right)(\boldsymbol{\rho}_{\mathtt{SZA}})$$

since the key k_j is uniformly sampled and used only once. Repeating this argument for each round, we can therefore obtain an efficient algorithm that acts only on registers $\mathtt{Z} \otimes \mathtt{A}$ and convinces the verifier with approximately the same probability as $\mathcal{A}$ recovers s. Note that this is a valid prover for the zero-knowledge protocol. This means that we can run the extractor on this prover to output the pre-image of x and thus break the one-wayness of f. □

8.5 Impossibility of Concurrent 2PQC

Here we state and prove the main theorem of this section.

Theorem 17. *Assume the existence of post-quantum OWFs. Then there exists a functionality $\mathcal{F}$ such that, for any two-party protocol Π to compute $\mathcal{F}$ there exists a polynomial $t = t(\lambda)$, a distribution $\mathcal{D}$, a function* secret*, and a polynomial-time adversary $\mathcal{A}$ such that:*

– *In a concurrent execution scheduled by $\mathcal{A}$ of t copies of Π with parties receiving inputs from the chosen from $d \leftarrow \mathcal{D}$, then $\mathcal{A}$ learns* $\mathsf{secret}(d)$ *with probability negligibly close to one.*

– *In an ideal execution where the parties get access to t copies of the ideal functionality $\mathcal{F}$ and receive inputs chosen from $d \leftarrow \mathcal{D}$, then any polynomial time adversary $\widetilde{\mathcal{A}}$ will only output* $\mathsf{secret}(d)$ *with negligible probability.*

Proof. Let f be a post-quantum OWF. We provide the description of the ideal functionality $\mathcal{F}$ below.

– Sender Input: A flag $j \in \{0,1,2,3\}$, a statement x (which is supposed to be an image of f), a witness w (which is supposed to be the pre-image satisfying $f(w) = x$), a string r, an index i, a state $\mathbf{e}$, two bit-strings $\widetilde{a}$ and $\widetilde{b}$, and two bits c and d.

– Receiver Input: A flag $j' \in \{0,1,2,3\}$, a statement x, and a qubit $\mathbf{x}$.

– Upon receiving the input from both parties, the functionality computes its output to the Receiver (while the sender does not receive anything) as follows:

  - If $j = j' = 0$: Check if $f(w) = x$ and return 1 if this is the case; Return 0 otherwise.
  - If $j = j' = 1$: Compute $X^{\widetilde{a}} Z^{\widetilde{b}} \mathsf{QGEncode}(i, r, \mathbf{e}, \mathbf{x})$.
  - If $j = j' = 2$: Return $(\widetilde{a}, \widetilde{b})$.
  - If $j = j' = 3$: Return (c, d).
  - If $j \neq j'$: Return $\perp$.

Let Π be a protocol to implement $\mathcal{F}$, we can derive protocols Π^{ZK} and Π^{Encode} for the zero knowledge and the input encoding functions, respectively. Let $\ell = \ell(\lambda)$ be the number of rounds needed for Π^{ZK} and let $\mathcal{G}$ be the (quantum) functionality as defined in the proof of Lem. 21. We assume that the adversary is given as input ℓ copies of the garbled circuits for the function computed by $\mathcal{G}$ (where the sender's inputs to the $\mathcal{G}$ functionality, which are all classical, are hardwired to the function description) and we set $t = (m+2n) \cdot \ell + 1$, where m is the the size (i.e., the number of qubits) of each garbled circuit.

Recall that each garbled circuit corresponds to a state $\widetilde{R}$. On the other hand, the sender of $\mathcal{F}$ is given as input the registers $\mathbf{e}_{1,i}$ and the strings r_i, for all $i \in [n]$, and the strings $\widetilde{x}$ and $\widetilde{z}$ corresponding to the encoding information for each garbled circuit. For each of the ℓ iterations, we assume that the sender behaves as follows. We only describe the inputs of the sender that influence the output, whereas for the other inputs we assume that the sender sends something arbitrary. The following actions are performed in sequence:

– First, it sends all of the $(i, r_i, \mathbf{e}_{1,i})$ to the functionality in sequence along with two random bit-strings (a_i, b_i), setting $j = 1$.

– Then, it sends all of the (a_i, b_i), setting $j = 2$.

– Finally, it sends all bits $(\widetilde{x}_i, \widetilde{z}_i)$ from the encoding in sequence, setting $j = 3$.

Given this protocol, we make the following claims.

– **Claim:** In the real world, there is an adversary that learns the secret with probability negligibly close to one.

The attack is identical to the one describe in Lem. 21, where the attacker just forwards the prover messages to the ideal functionality, except that it forwards them qubit-by-qubit to $\mathcal{F}$, setting $j = 1$. Afterwards it queries the functionality on input $j = 2$ and $j = 3$ for m queries, receiving the keys for the quantum one-time pad. This allows it to retrieve the encoded input and evaluate each quantum garbled circuit. By the correctness of the quantum garbling scheme, the output of this procedure is statistically close to the message of $\mathcal{G}$, and thus the attack succeeds with roughly the same probability.

– **Claim:** In the ideal world, no adversary can learn the secret with non-negligible probability.

We show this with a reduction to the same claim, except where the function implemented by $\mathcal{G}$ is queried as an oracle, which was already proven in Lem. 21. First observe that the parties need to agree on the flag $j = j'$ to obtain a non-$\perp$ output, and therefore we can assume without loss of generality that the attacker queries the functionality in the correct order.

The reduction proceeds as follows: For every message of the adversary with flags $j = j' = 1$, the reduction returns the first register of a series of freshly sampled maximally entangled states (one for each qubit of the encoding). The reduction stores the input qubit $\boldsymbol{\psi}_i$ sent be the adversary and, after n queries, it sends the adjoint state $\boldsymbol{\psi}$ to the ideal functionality $\mathcal{G}$. It receives the state $\boldsymbol{\phi}$ as output and runs

$$(\widetilde{E}_1, \ldots, \widetilde{E}_n, \widetilde{Q}) \leftarrow \mathsf{QGSim}(1^\lambda, \mathsf{par}, \boldsymbol{\phi}).$$

For each encoding $\widetilde{E}_i$, the reduction teleports it into the previously sampled maximally entangled state and sends the teleportation measurements with flag $j = j' = 2$. Then teleports also $\widetilde{Q}$ into the view of the adversary (same as done by the adaptive simulator) and sends the teleportation measurements with flag $j = j' = 3$. Note that, except for the fact that the garbled circuit is computed with a simulator, the reduction perfectly simulates the view of the adversary. By the security of the garbling scheme, the distribution induced by the reduction is computationally indistinguishable from the original one, and therefore any advantage of the adversary carries over to the settings where $\mathcal{G}$ is queried as an oracle, except for a negligible factor. By the proof of Lem. 21, we can therefore bound this to a negligible function.

□

9 Acknowledgment

We thank Kai-Min Chung for helpful discussions regarding [ACL21]. We also thank the CRYPTO'23 reviewers for their constructive feedback.

G.M. is partially funded by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM research hub under grant number 16KISK038, and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA – 390781972.

Work partially done when X.L. was at Rice University, supported by Nai-Hui Chia's NSF award FET-2243659.

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, Heidelberg, Germany. 2, 17

ACC$^+$21. Bar Alon, Hao Chung, Kai-Min Chung, Mi-Ying Huang, Yi Lee, and Yu-Ching Shen. Round efficient secure multiparty quantum computation with identifiable abort. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 436–466, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 2

ACL21. Prabhanjan Ananth, Kai-Min Chung, and Rolando L. La Placa. On the concurrent composition of quantum zero-knowledge. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 346–374, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. ii, 2, 3, 4, 6, 7, 8, 18, 19, 30, 32, 61, 67, 76

BBC$^+$93. Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, Mar 1993. 55

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, Heidelberg, Germany. 2, 49, 54

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, Heidelberg, Germany. 2

BD18. Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 370–390, Panaji, India, November 11–14, 2018. Springer, Heidelberg, Germany. 13, 17

BPS06. Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *47th Annual Symposium on Foundations of Computer Science*, pages 345–354, Berkeley, CA, USA, October 21–24, 2006. IEEE Computer Society Press. 1, 13, 14, 58

BW16. Anne Broadbent and Evelyn Wainewright. Efficient simulation for quantum message authentication. In Anderson C. A. Nascimento and Paulo Barreto, editors, *ICITS 16: 9th International Conference on Information Theoretic Security*, volume 10015 of *Lecture Notes in Computer Science*, pages 72–91, Tacoma, WA, USA, August 9–12, 2016. Springer, Heidelberg, Germany. 14, 56

BY22. Zvika Brakerski and Henry Yuen. Quantum garbled circuits. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 804–817, 2022. 14, 56

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. 1

CCLY22. 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, Heidelberg, Germany. 12, 15, 16, 35

CF01. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 1

CKL03. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. 1

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. 1

DGJ$^+$20. Yfke Dulek, Alex B. Grilo, Stacey Jeffery, Christian Majenz, and Christian Schaffner. Secure multi-party quantum computation with a dishonest majority. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 729–758, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. 2

DL09. Ivan Damgård and Carolin Lunemann. Quantum-secure coin-flipping and applications. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 52–69, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany. 2

DNS98. Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *30th Annual ACM Symposium on Theory of Computing*, pages 409–418, Dallas, TX, USA, May 23–26, 1998. ACM Press. 1

DNS12. Frédéric Dupuis, Jesper Buus Nielsen, and Louis Salvail. Actively secure two-party evaluation of any quantum operation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 794–811, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 2

Fei90. Uriel Feige. *Alternative models for zero knowledge interactive proofs*. PhD thesis, Ph. D. thesis, Weizmann Institute of Science, Rehovot, Israel, 1990. 1

FS90. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd Annual ACM Symposium on Theory of Computing*, pages 416–426, Baltimore, MD, USA, May 14–16, 1990. ACM Press. 1

GJ13. Vipul Goyal and Abhishek Jain. On concurrently secure computation in the multiple ideal query model. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 684–701, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. 1

GJO10. Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Password-authenticated session-key generation on the internet in the plain model. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 277–294, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 1

GLPV20. Sanjam Garg, Xiao Liang, Omkant Pandey, and Ivan Visconti. Black-box constructions of bounded-concurrent secure computation. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 87–107, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany. 2

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, Heidelberg, Germany. 2

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

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, Heidelberg, Germany. 13, 49, 54

HSS11. Sean Hallgren, Adam Smith, and Fang Song. Classical cryptographic protocols in a quantum world. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 411–428, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. 2

Kat07. Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany. 1

Lin03. Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *35th Annual ACM Symposium on Theory of Computing*, pages 683–692, San Diego, CA, USA, June 9–11, 2003. ACM Press. 1, 2

Lin04. Yehuda Lindell. Lower bounds for concurrent self composition. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 203–222, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 1

LN11. Carolin Lunemann and Jesper Buus Nielsen. Fully simulatable quantum-secure coin-flipping and applications. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11: 4th International Conference on Cryptology in Africa*, volume 6737 of *Lecture Notes in Computer Science*, pages 21–40, Dakar, Senegal, July 5–7, 2011. Springer, Heidelberg, Germany. 2

LPY22. Xiao Liang, Omkant Pandey, and Takashi Yamakawa. A new approach to post-quantum non-malleability. Cryptology ePrint Archive, Report 2022/907, 2022. `https://eprint.iacr.org/2022/907`. 2, 4, 11, 16

MPR06. Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th Annual Symposium on Foundations of Computer Science*, pages 367–378, Berkeley, CA, USA, October 21–24, 2006. IEEE Computer Society Press. 1

MU05. Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005. 20

OPP14. Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 17

Pas03. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. 1

Pas04. Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 232–241, Chicago, IL, USA, June 13–16, 2004. ACM Press. 2, 3, 19

PR03. Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *44th Annual Symposium on Foundations of Computer Science*, pages 404–415, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. 2

PS04. Manoj Prabhakaran and Amit Sahai. New notions of security: Achieving universal composability without trusted setup. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 242–251, Chicago, IL, USA, June 13–16, 2004. ACM Press. 1

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, Heidelberg, Germany. 13, 49, 54

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553, New York, NY, USA, October 17–19, 1999. IEEE Computer Society Press. 2

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. 2, 3, 19

WZ82. William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982. 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

A An Alternative Construction of 1-1 Simulation Sound Gadget

Toward showing the alternative construction of the 1-1 simulation-sound gadget, we first establish a property for general post-quantum non-malleable commitments in the following Lem. 22. This lemma considers a QPT adversary $\mathcal{M}_\lambda(\rho_\lambda)$ that can be partitioned into *two stages* $\mathcal{M}_\lambda = (\mathcal{M}^1_\lambda, \mathcal{M}^2_\lambda)$. The first adversary $\mathcal{M}^1_\lambda$ with auxiliary input ρ_λ participates in the MIM execution of a non-malleable commitment NMCom as in the standard definition of non-malleability (see Sec. 3.2). At the end of the execution, $\mathcal{M}^1_\lambda(\rho_\lambda)$ passes its internal state to the second adversary $\mathcal{M}^2_\lambda$. Additionally, $\mathcal{M}^2_\lambda$ also learns the value $\widetilde{m}$ committed in the right interaction by $\mathcal{M}^1_\lambda(\rho_\lambda)$ in the MIM execution of NMCom. The following Lem. 22 says that such an $\mathcal{M}^2_\lambda$ with the corresponding input cannot predict the message m (which is a single bit from $\{0,1\}$) committed by the left honest committer in the MIM execution where $\mathcal{M}^1_\lambda(\rho_\lambda)$ participates.

Lemma 22. *Assume* NMCom *is post-quantum non-malleable commitment that is statistically-binding. Then, for any two-stage QPT adversary* $\mathcal{M} = \{(\mathcal{M}^1_\lambda, \mathcal{M}^2_\lambda), \rho_\lambda\}_{\lambda\in\mathbb{N}}$, *it holds that*

$$\left| \Pr\left[\mathsf{NMPred}(\mathcal{M}^1_\lambda, \mathcal{M}^2_\lambda, \rho_\lambda) = 1\right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda),$$

where the probability is taken over the experiment $\mathsf{NMPred}(\mathcal{M}^1_\lambda, \mathcal{M}^2_\lambda, \rho_\lambda)$ *defined in the following Expr. 1.*

Experiment 1: Bit Prediction for Non-Malleable Commitments

1. *Run an MIM execution of* NMCom *with* $\mathcal{M}^1_\lambda(\rho_\lambda)$ *being the MIM adversary. That is,* $\mathcal{M}^1_\lambda(\rho_\lambda)$ *simultaneously participates in two instances of the* NMCom. *In one instance (dubbed the* left session*),* $\mathcal{M}^1_\lambda(\rho_\lambda)$ *plays the role of the receiver, talking with an honest committer* C *who commits to a* randomly sampled single bit $m \in \{0,1\}$*; In the other instance (dubbed the* right session*),* $\mathcal{M}^1_\lambda(\rho_\lambda)$ *plays the role of the committer, talking to an honest receiver* R. *Let* $\mathsf{OUT}_{\mathcal{M}^1_\lambda}$ *be the final state of* $\mathcal{M}^1_\lambda(\rho_\lambda)$ *at the end of this MIM execution. Let* $\widetilde{m}$ *denote the bit that* $\mathcal{M}^1(\rho_\lambda)$ *committed to in the right session (recall that* NMCom *is statistically binding).*
2. *Invoke* M^2_λ *on input* $(\mathsf{OUT}_{\mathcal{M}^1_\lambda}, \widetilde{m})$. $M^2_\lambda(\mathsf{OUT}_{\mathcal{M}^1_\lambda}, \widetilde{m})$ *output a value* m'.
3. **Decision:** *This experiment outputs 1 if and only if* $m' = m$. *That is, it outputs 1 if and only if* M^2_λ *successfully predicts the bit* m *committed by the left honest committer in the MIM execution in Step 1.*

Proof. First, we remark that the execution of Expr. 1 is not efficient. In particular, there is no efficient way to obtain the value $\widetilde{m}$ that is required as the input to M^2_λ. This makes Lem. 22 non-trivial. Otherwise (i.e., if Expr. 1 were efficient), Lem. 22 would follow straightforwardly from the computationally hiding property of NMCom. Nevertheless, we show in the following that Lem. 22 can be reduced to the non-malleability of NMCom, also in a rather straightforward manner.

To see that, notice that Step 1 of Expr. 1 is nothing but the MIM execution as we described when defining (ordinary) non-malleability (see Sec. 3.2). Therefore, from the non-malleability of NMCom, we know that no efficient distinguisher D_λ, on input $(\mathsf{OUT}_{\mathcal{M}^1_\lambda}, \widetilde{m})$, can tell if the value m committed by the left honest committer is 0 or 1, except for with negligible probability. Then, it is not hard to see that Lem. 22 holds, because otherwise, the machine $\mathcal{M}^2_\lambda$ can be used as a distinguisher D_λ to break the non-malleability of NMCom. Since this reduction is straightforward, we omit the details. $\square$

1-1 Simulation-Sound Gadget. We now present the alternative construction of the 1-1 simulation-sound gadget (in Prot. 9) and prove its security based on Lem. 22.

Protocol 9: An Alternative 1-1 Simulation-Sound Gadget

This protocol is between a sender (dubbed S) and a receiver (dubbed R); Both of them take a string $\mathsf{id} \in \{0,1\}^\lambda$ as the common input, denoting the ID associated with this execution.

1. S samples $a \xleftarrow{\$} \{0,1\}$ and commits to it using NMCom, where S and R use id as the ID (or tag) for this NMCom.

2. R samples $b \xleftarrow{\$} \{0,1\}$ and sends it to S.

The following lemma can be treated as the counterpart of Lem. 1 but w.r.t. Prot. 9. It shows that Prot. 9 enjoys the same 1-1 simulation soundness as Prot. 1.

Lemma 23. *Let* NMCom *be a post-quantum non-malleable commitment that is both statically binding and first-message binding (as we defined for Lem. 1). Assuming* $\mathsf{id} \neq \widetilde{\mathsf{id}}$*, it holds that*

$$\Pr_{\mathcal{T}\leftarrow \mathrm{MIM}}[\widetilde{a} = \widetilde{b} \mid a = b] = \frac{1}{2} \pm \mathsf{negl}(\lambda),$$

where we use the similar notation as in Lem. 1. That is, $\mathcal{T} \leftarrow \mathrm{MIM}$ *denotes the transcript resulted from the MIM execution of Prot. 9,* $\widetilde{a}$ *is the value committed in the right* NMCom *by* $\mathcal{A}$*,* $\widetilde{b}$ *is the value sent by* R *in the right, and* a *and* b *are those on the left.*

Proof. Similar as in the proof of Lem. 1, we again use the *first message of the left* NMCom as a pivot to divide all possible schedules into two mutually exclusive and collectively exhaustive types:

- **Type-1:** They are the schedules where the right NMCom *starts* after (or in parallel with) *the first messages of the left* NMCom.
- **Type-2:** They are the schedules where the right NMCom *starts* before *the first message of the left* NMCom.

We remark that Lem. 23 for **Type-2** schedules follows from exactly the same argument (i.e., due to the first-message binding property of NMCom) as in the proof of Lem. 1. In the following, we only need to focus one **Type-1** schedules.

First, it follows from the computational-hiding property of NMCom that

$$\Pr_{\mathcal{T}\leftarrow \mathrm{MIM}}[a = b] = \frac{1}{2} \pm \mathsf{negl}(\lambda). \tag{43}$$

We also claim that

$$\Pr_{\mathcal{T}\leftarrow \mathrm{MIM}}[\widetilde{a} = \widetilde{b}] = \frac{1}{2}. \tag{44}$$

This follows from the fact that the right honest receiver's $\widetilde{b}$ is sampled *independently* of $\mathcal{M}$'s bit $\widetilde{a}$.

Then, the following holds (we omit the "$\mathcal{T} \leftarrow \mathrm{MIM}$" in the subscript for simplicity):

$$\begin{aligned}
\Pr[a = b \mid \widetilde{a} = \widetilde{b}] &= \frac{\Pr[\widetilde{a} = \widetilde{b} \mid a = b] \cdot \Pr[a = b]}{\Pr[\widetilde{a} = \widetilde{b}]} && (45)\\
&= (1 \pm \mathsf{negl}(\lambda)) \cdot \Pr[\widetilde{a} = \widetilde{b} \mid a = b] && (46)\\
&= \Pr[\widetilde{a} = \widetilde{b} \mid a = b] \pm \mathsf{negl}(\lambda), && (47)
\end{aligned}$$

where Eq. (46) follows from Eq. (43) and (44).

Now, assume for contradiction that Lem. 23 does not hold. That is, there exist an adversary $\mathcal{M}_\lambda(\rho_\lambda)$ and a polynomial $\mathsf{poly}(\lambda)$ such that for infinitely many $\lambda \in \mathbb{N}$ it holds that

$$\left| \Pr_{\mathcal{T}\leftarrow \mathrm{MIM}}[\widetilde{a} = \widetilde{b} \mid a = b] - \frac{1}{2} \right| \geq \frac{1}{\mathsf{poly}(\lambda)}. \tag{48}$$

Inequality (48) together with Eq. (47) implies

$$\left| \Pr_{\mathcal{T}\leftarrow \mathrm{MIM}}[a = b \mid \widetilde{a} = \widetilde{b}] - \frac{1}{2} \right| \geq \frac{1}{\mathsf{poly}(\lambda)} \pm \mathsf{negl}(\lambda). \tag{49}$$

Then, we can use this adversary $\mathcal{M}_\lambda(\rho_\lambda)$ to break the security property of NMCom as specified by Lem. 22. To do that, we construct a two-state adversary as follows. Let $\mathcal{M}_\lambda$'s ρ_λ be the ρ_λ as required by Expr. 1; We define the $\mathcal{M}^1_\lambda(\rho_\lambda)$ as required by Expr. 1 to be the machine $\mathcal{M}_\lambda(\rho_\lambda)$ in the MIM execution of Prot. 9 *but truncated right after the right* NMCom *is sent.* Let ϕ_λ denote the current state of $\mathcal{M}_\lambda(\rho_\lambda)$ at this point. Note that this ϕ_λ will be the $\mathsf{OUT}_{\mathcal{M}^1_\lambda}$ that will be passed as an input to the $\mathcal{M}^2_\lambda$ (that we are about to construct).

We construct the required $\mathcal{M}^2_\lambda$ as follows:

- On input the residual state ϕ_λ and $\widetilde{a}$ (i.e., the value committed in the right-side NMCom by the $\mathcal{M}^1_\lambda(\rho_\lambda)$ defined about), $\mathcal{M}^2_\lambda$ continues the execution of Prot. 9 from the state ϕ_λ. Recall that this execution was truncated right after the right NMCom is sent. Therefore, if $\mathcal{M}^2_\lambda$ resumes the execution, then the MIM $\mathcal{M}_\lambda$ is expecting a value $\widetilde{b}$ sent by the right receiver R. At this point, $\mathcal{M}^2_\lambda$ sends $\widetilde{b} \coloneqq \widetilde{a}$, i.e., it uses the $\widetilde{a}$ from its second input as the $\widetilde{b}$ from the right honest receiver.
- **Output of $\mathcal{M}^2_\lambda$:** it outputs the bit b sent by the (resumed) MIM adversary $\mathcal{M}_\lambda$ in the left execution.

It is then straightforward to see that the experiment Expr. 1 executed using the $(\mathcal{M}^1_\lambda, \mathcal{M}^2_\lambda, \rho_\lambda)$ defined above is identical to the 1-1 MIM execution of Prot. 9 *conditioned on* $\widetilde{a} = \widetilde{b}$, with $\mathcal{M}_\lambda(\rho_\lambda)$ being the MIM adversary. Thus, the LHS of Inequality (49) is exactly the probability that the $\mathcal{M}^2_\lambda$ defined above correctly predicts the bit a committed to be the left honest committer in Expr. 1. Therefore, Inequality (49) contradicts Lem. 22, finishing the proof of the current Lem. 23. □

B Proving Claim 5

Intuition. The proof for this claim is similar to [ACL21], where the authors also need to prove that the $\mathsf{trapdoor}$ witness is available, i.e., there are enough "slots" (our $\mathsf{gadgets}$ in their term) are matching, when the hybrid needs to perform WI. In the following, we use $\Sigma \subseteq [Q]$ to denote the set of indices of the left sessions whose WI starts in block B_j.

Similar to [ACL21], we divide all the $\mathsf{gadgets}$ of the left sessions into two types:

- **Type 1: Matching by Rigging.** They are left $\mathsf{gadgets}$ that match because of the Watrous rewinding performed over blocks $\{B_1, \ldots, B_{j-1}\}$ (recall that in hybrid $H'_{j,1}$, the first $j-1$ blocks are performed using Watrous rewinding). For any left session $i \in \Sigma$, we denote the total number of such $\mathsf{gadgets}$ in the i-th left session by Rig_i. By our choice of parameters (i.e., L, ℓ_{gad}, etc.), we will show via a combinatorial argument that $\mathsf{Rig}_i \geq 3Q^4\lambda$ except for negligible probability.
- **Type 2: Matching by Luck.** They are left $\mathsf{gadgets}$ that are not "picked" by the Watrous rewinding procedure, but just happen to match. For any left session $i \in \Sigma$, we denote the total number of such $\mathsf{gadgets}$ by Luck_i. Recall that in our protocol, for the k-th gadget of the i-th left session, the $a_{i,k}$ in Step 2b is sampled uniformly at random, independent of the $b_{i,k}$ committed by $\mathcal{A}$. This means that each gadget of the i-th left session matches with probability exactly $1/2$, *if they are not* **Type-1**. From the first bullet, we know the total number of **Type-1** $\mathsf{gadgets}$ in the i-th left session is:[37]

$$\ell_{\mathsf{gad}} - \mathsf{Rig}_i = \ell_{\mathsf{gad}} - 3Q^4\lambda = 120Q^7\lambda - 3Q^4\lambda.$$

Therefore, the expectation of Luck_i is $60Q^7\lambda - \frac{3}{2}Q^4\lambda$. It then follows from Chernoff bound that $\mathsf{Luck}_i \geq 60Q^7\lambda - 2Q^4\lambda$ except for negligible probability.

It then follows from the above bounds for Rig_i and Luck_i (and an application of the union bound) that

$$\Pr\left[\exists i \in \Sigma ~~\text{s.t.}~~ \mathsf{Rig}_i + \mathsf{Luck}_i < 60Q^7\lambda + Q^4\lambda\right] \leq \mathsf{negl}(\lambda),$$

which means that for if the WI of a left session starts in block B_j, the total number of matching $\mathsf{gadgets}$ for that session must exceed the threshold $\mathsf{Th} = 60Q^7\lambda + Q^4\lambda$ (except for negligible probability). Thus, the $\mathsf{trapdoor}$ witness must be available for the WI of that session.

In the sequel, we formalize the above intuition. Recall that we need to prove them in the current hybrid $H'_{j,1}$. However, since the first $j-1$ blocks in $H'_{j,1}$ involves Watrous rewinding, it is not a good place to derive bounds for Rig_i and Luck_i. Thus, we will first show a claim (in Sec. B.1) which allows us to define (the counterparts of) Rig_i and Luck_i in the real MIM execution (i.e., hybrid REAL); This claim will tell us that the bounds derived in REAL serve just as good as those in $H'_{j,1}$, which will eventually allow us to finish the above argument.

[37] Note that $3Q^4\lambda$ is a lower bound for Rig_i. Thus, our counting here is conservative in the sense that we may potentially count **Type-1** $\mathsf{gadgets}$ (which matches for sure because of Watrous rewinding) as **Type-2** $\mathsf{gadgets}$ (which matches with probability 1/2), but not the other way around.

B.1 Moving to REAL

The current hybrid is $H'_{j,1}$. We write this j as j^* in this proof to emphasize this index and to avoid potential index conflicts.

As mentioned at the beginning, we will formally define the number Rig_i and Luck_i in hybrid $H'_{j^*,1}$ (which is the hybrid we care) and their counterparts in REAL (which will facilitate our computation). Due to some technical reasons (that will become clear later), we will not define Rig_i and Luck_i directly; Instead, we first define some intermediate random variables Σ, $X_{i,j}$, Θ_i, and $Z_{i,k}$ (in Algo. B.1), and then use them to define Rig_i and Luck_i.

Algorithm B.1: Random Variables in $H'_{j^*,1}$

Execute hybrid $H'_{j^*,1}$. This yields a schedule S. Let $\{B_1, \ldots, B_L\}$ be the partition associated with S. Recall that the first $j^* - 1$ blocks in $H'_{j^*,1}$ are performed using Watrous rewinding, where in each block B_j ($j \le j^* - 1$), it is possible that a non-matching left gadget is picked by the Watrous rewinding procedure (to make it match).

Random Variables: We define some random variables w.r.t. the execution of $H_{j^*,1}$:

- Let Σ denote the subset of $[Q]$ such that Σ is the collection of the indices of left sessions whose WI starts in block B_{j^*}.
- For any $i \in \Sigma$ and any $j \in [j^* - 1]$, let $X_{i,j}$ be a binary random variable defined to be 1 iff in block B_j, a gadget of the i-th left session is picked by Watrous rewinding.
- For any $i \in \Sigma$, let Θ_i denote the indices of the *first* $3Q^4\lambda$ $\mathsf{gadgets}$ that are picked for Watrous rewinding in the i-th left session. Note that if less than $3Q^4\lambda$ $\mathsf{gadgets}$ of the i-th left session are picked for Watrous rewinding, Θ_i will contain all the $\mathsf{gadgets}$ that are picked for Watrous rewinding in this session; In that case, $|\Theta_i|$ is strictly less than $3Q^4\lambda$.
- For any $i \in \Sigma$ and $k \in [\ell_{\mathsf{gad}}] \setminus \Theta_i$, let $Z_{i,k}$ be a binary random variable defined as follows:
  - **(Type-1 k):** If the k-th gadget of the i-th left session is picked for Watrous rewinding, let $Z_{i,k}$ be a Bernoulli random variable with $p = \frac{1}{2}$.

    Comment: This corresponds to the case where this k-th gadget is *picked* for Watrous rewinding in the i-th left session, but it is not collected in Θ_i, because Θ_i already contains $3Q^4\lambda$ elements.
  - **(Type-2 k):** Otherwise, let $Z_{i,k} = 1$ iff the k-th gadget of the i-th left session matches. That is, the $b_{i,k}$ committed in the k-th SBCom is equal to the $a_{i,k}$ committed in the k-th ENMC in the i-th left session .

    Comment: This corresponds to the case where this k-th gadget in the i-th left session is *not picked* picked for Watrous rewinding, but it simply matches by luck (when $Z_{i,k} = 1$).

We now define Rig_i and Luck_i using the Σ, $X_{i,j}$, Θ_i, and $Z_{i,k}$ defined in Algo. B.1. We let

$$\forall i \in \Sigma,\ \mathsf{Rig}_i := \begin{cases} 3Q^4\lambda & \text{if } \sum_{j \in [j^*-1]} X_{i,j} \ge 3Q^4\lambda \\ \sum_{j \in [j^*-1]} X_{i,j} & \text{if } \sum_{j \in [j^*-1]} X_{i,j} < 3Q^4\lambda \end{cases}, \tag{50}$$

and let

$$\forall i \in \Sigma,\ \mathsf{Luck}_i := \sum_{k \in [\ell_{\mathsf{gad}}] \setminus \Theta_i} Z_{i,k}. \tag{51}$$

Intuitively, Rig_i is the number of $\mathsf{gadgets}$ (of the i-th left session) that are made match by Watrous rewinding, and Luck_i is the number of $\mathsf{gadgets}$ (of the i-th left session) that are made match by luck. But our definition actually identifies some matching-by-rigging $\mathsf{gadgets}$ as matching-by-luck $\mathsf{gadgets}$, i.e., the k-th left gadget where k is **Type-2** (recall the definition of **Type-2** from Algo. B.1). Notice that such a counting method is conservative (see also Footnote 37) in the sense that $\mathsf{Rig}_i + \mathsf{Luck}_i$ serves as a *lower bound* for the total number of matching $\mathsf{gadgets}$ of the i-th left session in $H'_{j^*,1}$. That is,

$$\forall i \in \Sigma,\ \mathsf{Rig}_i + \mathsf{Luck}_i \le \text{ Total number of matching } \mathsf{gadgets} \text{ of the } i\text{-th left session.} \tag{52}$$

We now define the counterparts of Σ, $X_{i,j}$, Θ_i, and $Z_{i,k}$ in REAL.

Algorithm B.2: Procedure Pick regarding REAL

Execute the REAL game. This yields a schedule S. Let $\{B_1, \ldots, B_L\}$ be the partition associated with S. For this S, consider the following sub-procedure:

Procedure $\mathsf{SubPick}(S)$. Iterate for $j = 1$ to $j^* - 1$:

1. Let $T_j \subseteq [Q]$ be the indices of the left sessions that have at least one fully nested gadget in B_j; It is possible that $T_j = \emptyset$. In that case, directly move onto the next iteration;
2. Sample $i \xleftarrow{\$} T_j$ uniformly at random;
3. Sample uniformly at random a gadget of session i that is fully nested in B_j.

Random Variables: We define some random variables w.r.t. the above random procedure:

– Let Σ' denote the subset of $[Q]$ such that Σ' is the collection of the indices of left sessions whose WI starts in block B_{j^*}.

– For any $i \in \Sigma'$ and any $j \in [j^* - 1]$, let $X'_{i,j}$ be a binary random variable defined to be 1 iff in block B_j, a gadget of the i-th left session is sampled by the above procedure.

– For any $i \in \Sigma'$, let Θ'_i denote the indices of the *first* $3Q^4\lambda$ $\mathsf{gadgets}$ that are sampled by the above procedure in the i-th left session.

– For any $i \in \Sigma'$ and any $k \in [\ell_{\mathsf{gad}}] \setminus \Theta'_i$, let $Z'_{i,k}$ be a binary random variable defined as follows:

- **(Type-1 k):** If this k-th gadget of the i-th left session is picked by the above procedure, let $Z'_{i,k}$ be a Bernoulli random variable with $p = \frac{1}{2}$.
- **(Type-2 k):** Otherwise, let $Z'_{i,k} = 1$ iff the k-th gadget of the i-th left session matches.

On the Probability Space. We remark that in the proof, we may use of one the following two different probability spaces for the above random variables:

1. They can be defined as a result from the whole random procedure Pick, i.e., executing REAL to sample a schedule S and then executing $\mathsf{SubPick}(S)$. In this case, both the sampling of S and the execution of $\mathsf{SubPick}(S)$ contribute randomness.
2. They can also be defined as a result from $\mathsf{SubPick}(S)$ for a fixed S. In this case, the sampling of S does not contribute any randomness; The randomness used to run $\mathsf{SubPick}(S)$ constitutes the whole probability space.

When using these random variables, we will state explicitly which probability space to use.

Similarly, we define

$$\forall i \in \Sigma',\ \mathsf{Rig}'_i := \begin{cases} 3Q^4\lambda & \text{if } \sum_{j\in[j^*-1]} X'_{i,j} \geq 3Q^4\lambda \\ \sum_{j\in[j^*-1]} X'_{i,j} & \text{if } \sum_{j\in[j^*-1]} X'_{i,j} < 3Q^4\lambda \end{cases}, \tag{53}$$

and

$$\forall i \in \Sigma',\ \mathsf{Luck}'_i := \sum_{k\in[\ell_{\mathsf{gad}}]\setminus\Theta'_i} Z'_{i,k}, \tag{54}$$

where the probability is taken over the whole random procedure Pick shown by Algo. B.2.

Next, we prove a claim saying that $\mathsf{Rig}'_i + \mathsf{Luck}'_i$ can be used as a good estimation for $\mathsf{Rig}_i + \mathsf{Luck}_i$.

Claim 18. *For any number C, it holds that*

$$\Pr\left[\exists i \in \Sigma \text{ s.t. } \mathsf{Rig}_i + \mathsf{Luck}_i < C\right] = \Pr\left[\exists i \in \Sigma' \text{ s.t. } \mathsf{Rig}'_i + \mathsf{Luck}'_i < C\right] \pm \mathsf{negl}(\lambda),$$

where the LHS probability is taken over the execution of $H'_{j^*,1}$ shown in Algo. B.1 and the RHS probability is taken over the whole random procedure Pick *shown in Algo. B.2.*

Proof. It suffices to prove the following stronger claim:

$$\big(\Sigma, \{X_{i,j}\}_{i\in\Sigma, j\in[j^*-1]},\ \{\Theta_i\}_{i\in\Sigma},\ \{Z_{i,k}\}_{i\in\Sigma,k\in[\ell_{\mathsf{gad}}]\setminus\Theta_i}\big) \overset{\text{c}}{\approx} \big(\Sigma', \{X'_{i,j}\}_{i\in\Sigma', j\in[j^*-1]},\ \{\Theta'_i\}_{i\in\Sigma'},\ \{Z'_{i,k}\}_{i\in\Sigma',k\in[\ell_{\mathsf{gad}}]\setminus\Theta'_i}\big), \tag{55}$$

where the LHS random variables are defined w.r.t. the execution of $H'_{j^*,1}$ shown in Algo. B.1 and the RHS random variables are defined w.r.t. the random procedure Pick shown in Algo. B.2.

We first show the following indistinguishability:

$$\big(\Sigma, \{X_{i,j}\}_{i\in\Sigma, j\in[j^*-1]},\ \{\Theta_i\}_{i\in\Sigma}\big) \overset{\text{c}}{\approx} \big(\Sigma', \{X'_{i,j}\}_{i\in\Sigma', j\in[j^*-1]},\ \{\Theta'_i\}_{i\in\Sigma'}\big), \tag{56}$$

Previously, we proved that the output of REAL and $H'_{j^*,2}$ are computationally close. Particularly, it implies that the distributions of schedule S (i.e., how messages are arranged) yielded by them are computationally close. Moreover, notice that $(\Sigma', \{X'_{i,j}\}, \{\Theta'_i\})$ can be efficiently derived from the schedule S (i.e., how messages are ordered) of REAL; Similarly, $(\Sigma, \{X_{i,j}\}, \{\Theta_i\})$ can be efficiently derived from the schedule S of $H'_{j^*,1}$. Therefore, Eq. (56) holds.

Next, we argue that for any *fixed* $(\sigma, \{x_{i,j}\}_{i\in\sigma,j\in[j^*-1]}, \{\theta_i\}_{i\in\sigma})$, both $\{Z_{i,j}\}$ and $\{Z'_{i,j}\}$ are a sequence of independent Bernoulli random variables with $p=\frac{1}{2}$. Formally, for any fixed $(\sigma, \{x_{i,j}\}_{i\in\sigma,j\in[j^*-1]}, \{\theta_i\}_{i\in\sigma})$ in the intersection of the support of $(\Sigma, \{X_{i,j}\}_{i\in\Sigma, j\in[j^*-1]},\ \{\Theta_i\}_{i\in\Sigma})$ and $(\Sigma', \{X'_{i,j}\}_{i\in\Sigma', j\in[j^*-1]},\ \{\Theta'_i\}_{i\in\Sigma'})$, it holds that

$$\{Z_{i,k}\}_{k\in[\ell_{\mathsf{gad}}]\setminus\theta_i} \overset{\text{i.d.}}{=\!=} \{Z'_{i,k}\}_{k\in[\ell_{\mathsf{gad}}]\setminus\theta_i} \overset{\text{i.d.}}{=\!=} \{A_{i,k}\}_{k\in[\ell_{\mathsf{gad}}]\setminus\theta_i}, \tag{57}$$

where each $A_{i,k}$ is an independent Bernoulli random variable with $p=\frac{1}{2}$.

To prove Eq. (57), first notice that each $Z'_{i,k}$ is an independent Bernoulli random variable with $p=\frac{1}{2}$. This follows from the definition of $Z'_{i,k}$ in Algo. B.2:

- If the k is **Type-1** (as defined in Algo. B.2), then $Z'_{i,k}$ is simply an independent Bernoulli random variable with $p=\frac{1}{2}$.
- If the k is **Type-2** (as defined in Algo. B.2), then $Z'_{i,k}$ is defined to be 1 iff $b_{i,k}=a_{i,k}$. By our protocol design, $a_{i,k}$ is sampled by the honest prover (of the i-th left session) uniformly at random from $\{0,1\}$. Thus, the event "$b_{i,k}=a_{i,k}$" is also an independent Bernoulli random variable with $p=\frac{1}{2}$.

The same argument applies to the $Z_{i,k}$ in Algo. B.1 as well. The only different is that in $H'_{j^*,1}$, the first j^*-1 blocks are performed using Watrous rewinding. However, it is straightforward to see that this will not change the fact that "$a_{i,k}=b_{i,k}$ with probability 1/2" for the k-th gadget of the i-th left session that *is not picked by the Watrous procedure*, simply because for those $\mathsf{gadgets}$, $a_{i,k}$ is sampled by the honest prover uniformly at random from $\{0,1\}$ *even if conditioned on the performance of Watrous rewinding.*

Eq. (56) and (57) imply Eq. (55). To see that, notice that the sequence $\{A_{i,k}\}_{k\in[\ell_{\mathsf{gad}}]\setminus\theta_i}$ can be sampled given (as it is just a sequence of independent Bernoulli random variables with $p=\frac{1}{2}$). If a distinguisher $\mathcal{D}$ can break Eq. (55), then we can build a new distinguisher $\mathcal{D}'$ to break Eq. (56) by invoking $\mathcal{D}$ and sampling sequence $\{A_{i,k}\}_{k\in[\ell_{\mathsf{gad}}]\setminus\theta_i}$ by himself.

This finishes the proof of Claim 18. □

To lower bound the total number of matching $\mathsf{gadgets}$ in $H'_{j^*,1}$, it follows from Claim 18 that we only need to focus on the value Rig'_i and Luck'_i in the REAL game. In Appx. B.2 and Appx. B.3, we show lower bounds for these two values. After that, we finish the proof for Claim 5 in Appx. B.4

B.2 Lower Bound for Rig'_i

For any left session, we first bound the max number of its $\mathsf{gadgets}$ that can be fully nested in a block.

Claim 19. *For any left session, it can have at most $12Q^2$ $\mathsf{gadgets}$ fully nested in any single block.*

Proof. Recall from our parameter setting that $\ell_{\mathsf{gad}} = 120Q^7\lambda$ and $L=24Q^6\lambda$. Also recall that there are $T=2Q\cdot(\ell_{\mathsf{gad}}\cdot s_{\mathsf{gad}}+\mathit{\Gamma}_{\text{WI}})$ messages in the Q-Q concurrent MIM execution of our protocol, where $s_{\mathsf{gad}} = \mathit{\Gamma}_{\text{SBC}}+\mathit{\Gamma}_{\text{ENMC}}$ denotes the size of each gadget. If we let s_B denote the size of each block, then it holds that

$$s_B = \frac{T}{L} = \frac{2Q\cdot(120Q^7\lambda\cdot s_{\mathsf{gad}}+\mathit{\Gamma}_{\text{WI}})}{24Q^6\lambda} = 10Q^2\cdot s_{\mathsf{gad}} + \frac{\mathit{\Gamma}_{\text{WI}}}{12Q^5\lambda} \le 12Q^2\cdot s_{\mathsf{gad}},$$

which implies that each block can at most contain $12Q^2$ fully nested left $\mathsf{gadgets}$.

This finishes the proof of Claim 19. □

Claim 20. *Fix an S in the support of* Pick *shown in Algo. B.2. For any $i \in [Q]$, let N_i denote the number of blocks that have at least one fully nested* gadget *of the i-th left session in S. It holds that for all $i \in [Q]$, $N_i \geq 6Q^5\lambda - \mathit{\Gamma}_{\mathrm{WI}}$.*

Proof. For any $i \in [Q]$, let u_i denote the number of blocks that contain at least $2s_{\mathsf{gad}}$ messages of the i-th left session (recall that s_{gad} is the size of each gadget). Note that $u_i - \mathit{\Gamma}_{\mathrm{WI}}$ lower-bounds N_i.

Let $\{b_1, \ldots, b_{u_i}\}$ be the number of left-session-i messages contained in each of these u_i blocks. Let the number of left-session-i messages contained in the remaining $L - u_i$ blocks be denoted as $\{a_1, \ldots, a_{L-u_i}\}$. Notice that by definition, each a_k ($\forall k \in [L - u_i]$) block contains $< 2s_{\mathsf{gad}}$ messages of the i-th left session. Thus, it holds that

$$\sum_{k=1}^{L-u_i} a_k < (L - u_i) \cdot 2s_{\mathsf{gad}}. \tag{58}$$

Also, since the total number of messages contained in each block is s_B, it holds that

$$\sum_{k=1}^{u_i} b_k \leq u_i \cdot s_B. \tag{59}$$

Observe that the sum of these b_k's and a_k's is exactly the total number of messages of the i-th left session, which is exactly the round complexity of our protocol. That is,

$$\sum_{k=1}^{u_i} b_k + \sum_{k=1}^{L-u_i} a_k = \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}}. \tag{60}$$

Eq. (60) together with Inequalities (58) and (59) imply that:

$$u_i \cdot s_B + (L - u_i) \cdot 2s_{\mathsf{gad}} \geq \ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}},$$

which further implies that:

$$\begin{aligned}
u_i &\geq \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}} - 2s_{\mathsf{gad}}L}{s_B - 2s_{\mathsf{gad}}} \\
&= \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}} - 2s_{\mathsf{gad}}L}{\frac{2Q(\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}})}{L} - 2s_{\mathsf{gad}}} && (61) \\
&= \frac{L}{2Q} \cdot \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}} - 2s_{\mathsf{gad}}L}{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}} - 2s_{\mathsf{gad}} \cdot \frac{L}{2Q}} \\
&\geq \frac{L}{2Q} \cdot \frac{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}} - 2s_{\mathsf{gad}}L}{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}}} \\
&= \frac{L}{2Q} \cdot \left(1 - \frac{2s_{\mathsf{gad}}L}{\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}}}\right) \\
&\geq \frac{L}{2Q} \cdot \left(1 - \frac{2L}{\ell_{\mathsf{gad}}}\right) \\
&= \frac{L}{2Q} \cdot \left(1 - \frac{2}{5Q}\right) && (62) \\
&\geq \frac{L}{2Q} \cdot \left(1 - \frac{1}{2}\right) \\
&= 6Q^5\lambda && (63)
\end{aligned}$$

where Eq. (61) follows from our parameter setting of $s_B = \frac{2Q(\ell_{\mathsf{gad}} \cdot s_{\mathsf{gad}} + \mathit{\Gamma}_{\mathrm{WI}})}{L}$, Eq. (62) follows from our parameter setting of $L = 24Q^6\lambda$ and $\ell_{\mathsf{gad}} = 120Q^7\lambda$, and Eq. (63) follows from our parameter setting of $L = 24Q^6\lambda$.

As mentioned before, $u_i - \mathit{\Gamma}_{\text{WI}}$ lower-bounds N_i. It then follows from Eq. (63) that $N_i \geq 6Q^5\lambda - \mathit{\Gamma}_{\text{WI}}$. This finishes the proof of Claim 20.

□

Claim 21 (Matching by Rigging)**.** *It holds that*

$$\Pr\left[\exists i \in \mathit{\Sigma}' \ s.t. \sum_{j \in [j^*-1]} X'_{i,j} < 3Q^4\lambda\right] \leq \mathsf{negl}(\lambda), \tag{64}$$

where both the expectation and the probability is taken over the whole random procedure Pick *shown in Algo. B.2.*

Proof. We emphasize that probability space for Inequality (64) is the whole random procedure Pick. To prove this claim, it suffices to show the following stronger claim: *For any fixed S* (resulted from the execution of REAL), Inequality (64) hold when the probability space is defined to be the sub-random-procedure $\mathsf{SubPick}(S)$ shown in Algo. B.2. (That is, the S is given as an a-priori fixed input; The randomness used to generate S does not contribute to the above expectation and probability.) In the sequel, we show the argument for a fix S, using the sub-random-procedure $\mathsf{SubPick}(S)$ as the probability space.

Note that if S is fixed, the $\mathit{\Sigma}'$ is also fixed to some value σ'. For $i \in \sigma'$ and $j \in [j^*-1]$, let $b_{i,j}$ be a binary indicator defined to be 1 iff there exists a fully nested left-session-i gadget in block B_j. We claim that

$$\forall i \in \sigma', \ \textstyle\sum_{j \in [j^*-1]} b_{i,j} \geq N_i - 12Q^2 \geq 6Q^5\lambda - \mathit{\Gamma}_{\text{WI}} - 12Q^2, \tag{65}$$

where N_i is defined in Claim 20. To see why the first "$\geq$" in Inequality (65) holds, notice that the summation of $b_{i,j}$ is over $j \in [j^*-1]$; If $i \in \sigma'$, we know by the definition of σ' that the WI of the i-th left session starts in block B_{j^*}, which means all of its gadgets are finished before the end of B_{j^*}. But it is possible that some of it gadgets are fully nested *within* B_{j^*}. By Claim 19, we know that there are at most $12Q^2$ left gadets that could be fully nested in a single block. Thus, $N_i - 12Q^2$ is a lower bound for the number of fully nested gadgets in blocks $\{B_1, \ldots, B_{j^*-1}\}$ (i.e., excluding the fully nested left gadgets in block B_{j^*}). (The last "$\geq$" in Inequality (65) follows from Claim 20.)

Since the random procedure described in Algo. B.2 picks the session index i *uniformly at random* in Step 2 (where $|T_j| \leq Q$), it holds that

$$\forall i \in \sigma', \ \mathbb{E}\big[\textstyle\sum_{j \in [j^*-1]} X'_{i,j}\big] \geq \textstyle\sum_{j \in [j^*-1]} b_{i,j} \cdot \frac{1}{Q}, \tag{66}$$

where the expectation is taken over the sub-random-procedure $\mathsf{SubPick}(S)$ for a fixed S.

It then follows from Inequalities (65) and (66) that

$$\forall i \in \sigma', \ \mathbb{E}\big[\textstyle\sum_{j \in [j^*-1]} X'_{i,j}\big] \geq 6Q^4\lambda - \frac{\mathit{\Gamma}_{\text{WI}}}{Q} - 12Q, \tag{67}$$

where the expectation is taken over the sub-random-procedure $\mathsf{SubPick}(S)$ for a fixed S. (Recall that $\mathit{\Gamma}_{\text{WI}}$ is a constant.)

To prove Claim 21, first notice that the values X'_{i,j_1} and X'_{i,j_2} are independently distributed for any $j_i, j_2 \in [j^*-1]$. This is because the probability is taken over the procedure $\mathsf{SubPick}(S)$ for a *fixed* S. Once S is fixed, how left gadgets are nested in blocks is also a fixed fact. The result of $\mathsf{SubPick}(S)$ does not correlate between different bocks. Therefore, we can apply Chernoff inequality to Inequality (67), which implies

$$\forall i \in \sigma', \ \Pr\left[\sum_{j \in [j^*-1]} X'_{i,j} < 3Q^4\lambda\right] \leq \mathsf{negl}(\lambda), \tag{68}$$

where the expectation is taken over the sub-random-procedure $\mathsf{SubPick}(S)$ for a fixed S.

Then, notice that $|\sigma'|$ (which is $\leq Q$) is a polynomial of the security parameter λ. Thus, an application of the union bound to Inequality (68) yields that

$$\Pr\left[\exists i \in \sigma' \text{ s.t.} \sum_{j \in [j^*-1]} X'_{i,j} < 3Q^4\lambda\right] \leq \mathsf{negl}(\lambda). \tag{69}$$

where the probability is taken over the sub-random-procedure $\mathsf{SubPick}(S)$ for a fixed S (thus a fixed σ').

Since Inequality (69) holds for any schedule S that could possibly come from REAL, Inequality (69) implies Claim 21 where the randomness of sampling S contributes to the probability space.

This finishes the proof of Claim 21. □

B.3 Lower Bound for Luck'_i

Claim 22 (Matching by Luck). *Let $Z'_{i,k}$ and Θ'_i be as defined in Algo. B.2. It holds that*

$$\Pr\Big[\exists i \in \Sigma' \text{ s.t. } \textstyle\sum_{k\in[\ell_{\mathsf{gad}}]\setminus\Theta'_i} Z'_{i,k} < 60Q^7\lambda - 2Q^4\lambda\Big] \le \mathsf{negl}(\lambda),$$

where the probability is taken over the whole random procedure Pick *as defined by Algo. B.2.*

Proof. Similar as in the proof of Claim 21, we first show the argument for a fix S, using the sub-random-procedure $\mathsf{SubPick}(S)$ as the probability space. This also fix Σ to some σ'.

First, recall (from the proof of Claim 18) that $Z'_{i,k}$'s are independent Bernoulli random variables with $p = \frac{1}{2}$. Thus, for any $i \in \sigma'$, it holds that

$$\begin{aligned}
\mathbb{E}[\textstyle\sum_{k\in[\ell_{\mathsf{gad}}]\setminus\Theta'_i} Z'_{i,k}] &= \frac{1}{2}\big|[\ell_{\mathsf{gad}}] \setminus \Theta'_i\big| \\
&= \frac{1}{2}\big(\ell_{\mathsf{gad}} - |\Theta'_i|\big) \\
&= \frac{1}{2}\big(120Q^7\lambda - |\Theta'_i|\big) && (70)\\
&\ge \frac{1}{2}\big(120Q^7\lambda - 3Q^4\lambda\big) && (71)\\
&= 60Q^7\lambda - \frac{3}{2}Q^4\lambda, && (72)
\end{aligned}$$

where Eq. (70) follows from our parameter setting that $\ell_{\mathsf{gad}} = 120Q^7\lambda$, and Eq. (71) follows by definition that $|\Theta'_i| \le 3Q^4\lambda$.

Then, an application of Chernoff bound to Eq. (72) followed by an application of the union bound yields:

$$\Pr\Big[\exists i \in \sigma' \text{ s.t. } \textstyle\sum_{k\in[\ell_{\mathsf{gad}}]\setminus\Theta'_i} Z'_{i,k} < 60Q^7\lambda - 2Q^4\lambda\Big] \le \mathsf{negl}(\lambda), \tag{73}$$

where the probability is taken over the sub-random-procedure $\mathsf{SubPick}(S)$ for a fixed S (thus a fixed σ').

Since Inequality (73) holds for any schedule S that could possibly come from REAL, Inequality (73) implies Claim 22 where the randomness of sampling S contributes to the probability space.

This finishes the proof of Claim 22. □

B.4 Finishing the Proof

In the following, all the probabilities are taken over the whole random procedure Pick shown in Algo. B.2.

Claim 21 together with the definition of Rig'_i in Eq. (53) imply that

$$\Pr\big[\exists i \in \Sigma' \text{ s.t. } \mathsf{Rig}'_i < 3Q^4\lambda\big] \le \mathsf{negl}(\lambda). \tag{74}$$

Claim 22 together with the definition of Lick'_i in Eq. (54) imply that

$$\Pr\big[\exists i \in \Sigma' \text{ s.t. } \mathsf{Luck}'_i < 60Q^7\lambda - 2Q^4\lambda\big] \le \mathsf{negl}(\lambda). \tag{75}$$

Then, it holds that

$$\Pr\left[\forall i \in \Sigma',\ \mathsf{Rig}'_i + \mathsf{Luck}'_i \ge 60Q^7\lambda + Q^4\lambda\right]$$

$$\begin{aligned}
&\geq \Pr\left[\forall i \in \Sigma',\ \left(\mathsf{Rig}'_i \geq 60Q^7\lambda - 2Q^4\right) \wedge \left(\mathsf{Luck}'_i \geq 3Q^4\lambda\right)\right] \\
&= 1 - \Pr\left[\exists i \in \Sigma' \text{ s.t. } \left(\mathsf{Rig}'_i < 60Q^7\lambda - 2Q^4\right) \vee \left(\mathsf{Luck}'_i < 3Q^4\lambda\right)\right] \\
&= 1 - \Pr\left[\left(\exists i \in \Sigma' \text{ s.t. } \mathsf{Rig}'_i < 60Q^7\lambda - 2Q^4\right) \vee \left(\exists i \in \Sigma' \text{ s.t. } \mathsf{Luck}'_i < 3Q^4\lambda\right)\right] \\
&\geq 1 - \Pr\left[\exists i \in \Sigma' \text{ s.t. } \mathsf{Rig}'_i < 60Q^7\lambda - 2Q^4\right] - \Pr\left[\exists i \in \Sigma' \text{ s.t. } \mathsf{Luck}'_i < 3Q^4\lambda\right] && (76)\\
&\geq 1 - \Pr\left[\exists i \in \Sigma' \text{ s.t. } \mathsf{Rig}'_i < 60Q^7\lambda - 2Q^4\right] - \mathsf{negl}(\lambda) && (77)\\
&\geq 1 - \mathsf{negl}(\lambda) && (78)
\end{aligned}$$

where Inequality (76) follows from the union bound, Inequality (77) follows from Inequality (75), and Inequality (78) follows from Inequality (74).

Inequality (78) implies the following

$$\Pr\left[\exists i \in \Sigma' \text{ s.t. } \mathsf{Rig}'_i + \mathsf{Luck}'_i < 60Q^7\lambda + Q^4\lambda\right] \leq \mathsf{negl}(\lambda), \quad (79)$$

where the probability is taken over the whole random procedure Pick as defined by Algo. B.2.

Inequality (79) and Claim 18 together imply

$$\Pr\left[\exists i \in \Sigma \text{ s.t. } \mathsf{Rig}_i + \mathsf{Luck}_i < 60Q^7\lambda + Q^4\lambda\right] \leq \mathsf{negl}(\lambda), \quad (80)$$

where the probability is taken over the execution of $H'_{j^*,1}$ as defined in Algo. B.1.

Finally, recall from Inequality (52) that for any $i \in \Sigma$ (i.e., the i-th left session whose WI starts in block B_{j^*}), the value $\mathsf{Rig}_i + \mathsf{Luck}_i$ lower bounds the total number of matching gadgets of the i-th left session. Therefore, Inequality (80) says exactly what we want—If the WI of some i-th left session *starts* in block B_{j^*}, then the trapdoor witness for that session must be available, except for negligible probability.

This eventually finishes the proof of Claim 5.

C Proving Indistinguishability of Lem. 11

We first define two intermediate hybrids, which are identical to $H_{j,1}$ and $H_{j,2}$ but truncated at the end of block B_j:

– **Hybrid $H^*_{j,1}$:** Proceed identically as $H_{j,1}$ until the end of block B_j. Then, measure every registers except for the `Adv` register, and output the contents in registers `Adv` and `Tran`.

– **Hybrid $H^*_{j,2}$:** Proceed identically as $H_{j,2}$ until the end of block B_j. Then, measure every registers except for the `Adv` register, and output the contents in registers `Adv` and `Tran`.

The proof of $\mathsf{OUT}(H_{j,1}) \overset{c}{\approx} \mathsf{OUT}(H_{j,2})$ goes in two steps: (i) We first reduce it to proving $\mathsf{OUT}(H^*_{j,1}) \overset{c}{\approx} \mathsf{OUT}(H^*_{j,2})$; (ii) We then prove $\mathsf{OUT}(H^*_{j,1}) \overset{c}{\approx} \mathsf{OUT}(H^*_{j,2})$.

For Step (i). Notice that after block B_j, hybrids $H_{j,1}$ and $H_{j,2}$ use identical strategies to finish the remaining blocks $\{B_{j+1}, \ldots, B_L\}$. Thus, their output should be computationally indistinguishable if their states at the end of B_j are computationally indistinguishable. There is a caveat: The statement $\mathsf{OUT}(H^*_{j,1}) \overset{c}{\approx} \mathsf{OUT}(H^*_{j,2})$ is only about the output of these two hybrids. That is, at the end of block B_j, the contents in `Adv` and `Tran` are indistinguishable, while the contents in other registers could contain information that distinguish these two hybrids. But it is straightforward to see that the contents in other registers do not contribute to the execution after block B_j, except for the $\otimes_{i=1}^{Q}\mathtt{RT}_i$ register, which determines if the remaining blocks should use the real witness or the trapdoor witness for the left sessions.

Therefore, given $\mathsf{OUT}(H^*_{j,1}) \overset{c}{\approx} \mathsf{OUT}(H^*_{j,2})$, the only reason that $\mathsf{OUT}(H_{j,1})$ and $\mathsf{OUT}(H_{j,2})$ are distinguishable is because the trapdoor witness for some i-th left session is unavailable in $H_{j,1}$ (i.e., $|0\rangle_{\mathtt{RT}_i}$) but becomes available in $H_{k,2}$ (i.e., $|1\rangle_{\mathtt{RT}_i}$) *and is used by $H_{k,2}$ to perform the corresponding* WI. Indeed, this is possible: Compared with $H_{j,1}$, $H_{j,2}$ could potentially make one more left gadget match—the one fully nested in B_j—because it performs Watrous rewinding for block B_j. It is possible that this one more match suddenly makes the trapdoor witness available for some i-th left session in $H_{j,2}$.

However, it is easy to see that such a switch of witness will not affect the indistinguishability between $\mathsf{OUT}(H_{j,1})$ and $\mathsf{OUT}(H_{j,2})$, because otherwise, one can construct an adversary that breaks the witness indistinguishability of the Stage 3 WI of the i-th left session. Since this argument is standard, we omit the details.

For Step (ii). In the following, we show that $\mathsf{OUT}(H^*_{j,1}) \overset{\text{c}}{\approx} \mathsf{OUT}(H^*_{j,2})$.

Recall that the state at the end of block B_{j-1} is identical in both $H^*_{j,1}$ and $H^*_{j,2}$, which is $|\psi_{j-1}\rangle = W_{j-1} \cdots W_0 |\psi_0\rangle$. Then, $H^*_{j,1}$ will apply the U'_j defined in Step 2 of $H_{j,1}$, while $H^*_{j,2}$ will apply the W_j defined in Step 2 of $H_{j,2}$ to $|\psi_{j-1}\rangle$.

Without loss of generality, $U'_j |\psi_{j-1}\rangle$ can be written in the following way for some $q \in [0,1]$:

$$U'_j |\psi_{j-1}\rangle = \sqrt{q} \cdot |\psi_j^{\mathsf{nogad}}\rangle + \sqrt{1-q} \cdot |\psi_j^{\mathsf{gad}}\rangle, \tag{81}$$

where

- $|\psi_j^{\mathsf{nogad}}\rangle$ is a superposition corresponding to all the executions where there is no fully nested left gadget in block B_j. This is defined on all the registers. This state can be further decomposed as

$$|\psi_j^{\mathsf{nogad}}\rangle = \frac{1}{\sqrt{2}} \cdot |\psi_j^{\mathsf{nogad},0}\rangle |0\rangle_{\mathtt{W}_j} + \frac{1}{\sqrt{2}} \cdot |\psi_j^{\mathsf{nogad},1}\rangle |1\rangle_{\mathtt{W}_j}, \tag{82}$$

  Notice that the amplitudes for both branch is $1/\sqrt{2}$ because, conditioned on there is no fully nested left gadget in B_j, the value in register $\mathtt{W}_j$ is defined to be a bit c_j sampled uniformly at random (recall it from Step 2c).

- $|\psi_j^{\mathsf{gad}}\rangle$ is a superposition corresponding to all the executions where there is at least one fully nested left gadget in block B_j. This state can be further decomposed as

$$|\psi_j^{\mathsf{gad}}\rangle = \frac{1}{\sqrt{2}} \cdot |\psi_j^{\mathsf{gad},0}\rangle |0\rangle_{\mathtt{W}_j} + \frac{1}{\sqrt{2}} \cdot |\psi_j^{\mathsf{gad},1}\rangle |1\rangle_{\mathtt{W}_j}. \tag{83}$$

  Notice that the amplitudes for both branch is $1/\sqrt{2}$ because, conditioned on there exits at least one fully nested left gadget, a randomly selected one from them matches (aka $\mathtt{W}_j$ being set to 0) with probability exactly $1/2$ (because the a_k in Step 2b of our protocol is sampled uniformly).

Eq. (81) together with Eq. (82) and (83) imply that

$$U'_j |\psi_{j-1}\rangle = \frac{1}{\sqrt{2}} \cdot |\psi_j^0\rangle |0\rangle_{\mathtt{W}_j} + \frac{1}{\sqrt{2}} \cdot |\psi_j^1\rangle |1\rangle_{\mathtt{W}_j}, \tag{84}$$

where we used the following notations:

- $|\psi_j^0\rangle \coloneqq \sqrt{q} \cdot |\psi_j^{\mathsf{nogad},0}\rangle + \sqrt{1-q} \cdot |\psi_j^{\mathsf{gad},0}\rangle$. This state corresponds to the superposition over all registers (excluding $\mathtt{W}_j$) such that either one of the following two conditions are satisfied: (i) there is no fully nested left gadget in block B_j and the random coin c_j tosses to 0 (recall it from Step 2c), or (ii) the chosen left gadget (fully nested in B_j) matches.
- $|\psi_j^1\rangle \coloneqq \sqrt{q} \cdot |\psi_j^{\mathsf{nogad},1}\rangle + \sqrt{1-q} \cdot |\psi_j^{\mathsf{gad},1}\rangle$. This state corresponds to the superposition over all registers (excluding $\mathtt{W}_j$) such that either one of the following two conditions are satisfied: (i) there is no fully nested left gadget in block B_j but the random coin c_j tosses to 1 (recall it from Step 2c), or (ii) the chosen left gadget (fully nested in B_j) does not match.

Applying Watrous Lemma. Now, recall that the $U'_j |\psi_{j-1}\rangle$ shown in Eq. (84) is the state at the end of B_j in hybrid $H^*_{j,1}$. In contrast, the counterpart state in hybrid $H^*_{j,2}$ is $W_j |\psi_{j-1}\rangle$, where by definition $W_j = \mathsf{Amplifier}(U'_j)$ with $\mathtt{W}_j$ playing the role of the $\mathsf{Watrous\ control\ register}$ (recall it from Step 2). It then follows from an application of Lem. 6 (setting $p_1 = 0.5$ and $p_0 = 0.49$) that

$$W_j |\psi_{j-1}\rangle \overset{\text{s}}{\approx} |\psi_j^0\rangle |0\rangle_{\mathtt{W}_j}, \tag{85}$$

which means that in hybrid $H^*_{j,2}$, the state obtained after the execution of block B_j is exponentially close in trace distance to the state $|\psi_j^0\rangle |0\rangle_{\mathtt{W}_j}$.

Indistinguishability Between $\mathsf{OUT}(H^*_{j,1})$ and $\mathsf{OUT}(H^*_{j,2})$. Recall that our eventual goal is to show $\mathsf{OUT}(H^*_{j,1}) \overset{c}{\approx} \mathsf{OUT}(H^*_{j,2})$, which will hold as long as the following holds

$$\mathrm{Tr}_{\mathtt{W}_j}\left(W_j \ket{\psi_{j-1}}\bra{\psi_{j-1}} W_j^\dagger\right) ~\overset{c}{\approx}~ \mathrm{Tr}_{\mathtt{W}_j}\left(U'_j \ket{\psi_{j-1}}\bra{\psi_{j-1}} U'^\dagger_j\right) \tag{86}$$

To show Eq. (86), due to Eq. (84) and (85), it suffices to show the following:

$$\ket{\psi^0_j}\bra{\psi^0_j} ~\overset{c}{\approx}~ \frac{1}{2}\cdot\ket{\psi^0_j}\bra{\psi^0_j} + \frac{1}{2}\cdot\ket{\psi^1_j}\bra{\psi^1_j}, \tag{87}$$

To see why Eq. (87) holds, first recall that for any two quantum states $\rho_0 \overset{c}{\approx} \rho_1$ and any constant $p \in [0,1]$,[38] it holds that

$$\rho_0 = p\cdot\rho_0 + (1-p)\cdot\rho_0 ~\overset{c}{\approx}~ p\cdot\rho_0 + (1-p)\cdot\rho_1.$$

Thus, to show Eq. (87), it suffices to show that

$$\ket{\psi^0_j}\bra{\psi^0_j} ~\overset{c}{\approx}~ \ket{\psi^1_j}\bra{\psi^1_j}. \tag{88}$$

Eq. (88) simply follows from the computationally hiding property of ENMC. Since this is a simple and standard reduction, we omit the details (see [ACL21, Claim 29] for an example).

[38] Actually, it suffices as long as p is a noticeable function on the security parameter λ.