The Round Complexity of Black-Box Post-Quantum Secure Computation

Rohit Chatterjee[1], Xiao Liang[2], Omkant Pandey[3], and Takashi Yamakawa[4]

[1] National University of Singapore, Singapore
rochat@nus.edu.sg
[2] The Chinese University of Hong Kong, Hong Kong
xiaoliang@cuhk.edu.hk
[3] Stony Brook University, USA
omkant@cs.stonybrook.edu
[4] NTT Social Informatics Laboratories, Japan
takashi.yamakawa@ntt.com

Abstract. We study the round-complexity of secure multi-party computation (MPC) in the post-quantum regime where honest parties and communication channels are classical but the adversary can be a quantum machine. Our focus is on the *fully* black-box setting where both the construction as well as the security reduction are black-box in nature. In this context, Chia, Chung, Liu, and Yamakawa [FOCS'22] demonstrated the infeasibility of achieving standard simulation-based security within constant rounds, unless $\mathbf{NP} \subseteq \mathbf{BQP}$. This outcome leaves crucial feasibility questions unresolved. Specifically, it remains unknown whether black-box constructions are achievable within polynomial rounds; additionally, the existence of constant-round constructions with respect to *ε-simulation*, a relaxed yet useful alternative to the standard simulation notion, remains unestablished.

This work provides positive answers to the aforementioned questions. We introduce the first black-box construction for post-quantum MPC in polynomial rounds, from the minimal assumption of post-quantum semi-honest oblivious transfers. In the two-party scenario, our construction requires only $\omega(1)$ rounds. These results have already found application in the oracle separation between classical-communication quantum MPC and $\mathbf{P} = \mathbf{NP}$ in the recent work of Kretschmer, Qian, and Tal [STOC'25].

As for ε-simulation, Chia, Chung, Liang, and Yamakawa [CRYPTO'22] resolved the issue for the two-party setting, leaving the general multi-party setting as an open question. We complete the picture by presenting the first black-box and constant-round construction in the multi-party setting. Our construction can be instantiated using various standard post-quantum primitives including lossy public-key encryption, linearly homomorphic public-key encryption, or dense cryptosystems.

En route, we obtain a black-box and constant-round post-quantum commitment that achieves a weaker version of the standard 1-many non-malleability, from the minimal assumption of post-quantum one-way functions. Besides its utility in our post-quantum MPC construction, this commitment scheme also reduces the assumption used in the lower bound of quantum parallel repetition recently established by Bostanci, Qian, Spooner, and Yuen [STOC'24]. We anticipate that it will find more applications in the future.

Keywords: Multi-Party Computation · Post-Quantum · Non-Malleability · Black-Box · Round Complexity

Table of Contents

1 Introduction

Secure multi-party computation (MPC) allows two or more mutually distrustful parties to compute any functionality without compromising the privacy of their inputs [Yao86, GMW87]. We study foundational questions pertaining to the efficiency of secure multiparty computation in the *post-quantum* regime where honest parties and communication channels are classical but the adversary can be a quantum machine. We focus on two specific efficiency criteria: (1) the round-complexity, and (2) the black-box nature of the protocols. We are only concerned with general-purpose protocols in this work, i.e., protocols that can compute any well-defined multiparty functionality.

The black-box nature of the protocols manifests itself in at least two ways. First, the MPC protocol is said to have a *black-box construction*, if it only relies on the input-output behaviour of the underlying cryptographic primitives/assumptions. That is, the description of the MPC protocol is independent of the *implementation* level details of the underlying cryptographic primitives. This ensures that the efficiency of the protocol does not change with the implementation details of the underlying primitives. Moreover, such constructions remain valid even if the building-block primitives are based on a physical object such as a noisy channel or tamper-proof hardware [Wyn75, CK88, GLM+04].

Second, the MPC protocol is said to have a *black-box security-proof (or reduction)* if the security proof uses the adversary only as a black-box (i.e., only relies on its input/output functionality). We are concerned with MPC protocols that are *fully* black-box [IR89, RTV04], i.e., they have a black-box construction as well as a black-box reduction to the underlying cryptographic primitives. Protocols that admit black-box reductions are often simpler and tend to result in more efficient implementations.

The complexity of black-box MPC protocols is well understood in the classical setting, resulting in fully black-box constructions in a *constant* number of rounds under standard polynomial hardness assumptions [Goy11], obtained after a long sequence of works in this direction [IKLP06, IKOS07, Hai08, IPS08, PW09, CDMW09, Wee10].

However, these questions are wide open in the post-quantum MPC (PQ-MPC) setting where honest parties and communication channels are still classical but the adversary is allowed to be a quantum machine. This is in part due to the fact that classical techniques for performing simulation and extraction in MPC and Zero-Knowledge protocols rarely work when the adversary is a quantum machine. In fact, Chia, Chung, Liu, and Yamakawa [CCLY22b] recently showed that standard (expected) polynomial-time black-box simulation is impossible to achieve by constant-round constructions in the post-quantum setting (unless $\mathbf{NP} \subseteq \mathbf{BQP}$). Indeed, this impossibility holds even in scenarios where honest parties have access to quantum capabilities [CCLL24]. These strong results still leave glaring feasibility questions unresolved, which we discuss next.

Full simulation but non-constant rounds: In this regime, all known constructions achieving full (or standard) simulation [ABG+21, LPY23a, GLM23][5] make extensive use of non-black-box techniques. Indeed, [ABG+21, LPY23a] even achieve constant rounds by relying on non-black-box simulation. However, no results are known if we insist on black-box constructions, *even* for the two-party setting.[6] This raises the following question:

[5] Note that [LPY23b] and [LPY23a] refer to the same paper. We include two separate bibliography entries because certain lemmas appear exclusively in the arXiv version [LPY23b] but not in the conference version [LPY23a], and we occasionally need to cite them specifically.

[6] We note that fully black-box constructions exist if the honest parties are allowed to leverage quantum power (e.g., [BCKM21, GLSV21]). However, this falls outside the scope of our focus on *post-quantum* protocols.

Question 1: Do there exist black-box constructions of post-quantum 2PC (and MPC) with full simulation (in more than constant number of rounds)?

Relaxed simulation in constant rounds: The everpresent desire for constant-round secure protocols has prompted exploration of alternative notions such as *with ε-simulation*, which is a relaxed form of standard simulation-based security that allows for an arbitrarily small noticeable simulation error ε. This is an extensively well-studied notion in the literature [DNRS99, JKKR17, BKP19] that implies other important security notions — e.g., ε-zero-knowledge protocols imply witness indistinguishability [FS90] and ε-simulatable MPCs imply input-indistinguishable computation [MPR06]. In this ε-simulation regime, the recent work of [CCLY22a] made initial progress by presenting a constant-round fully black-box protocol for the two-party setting. However, obtaining similar results in the *multi-party setting* has remained an unsolved challenge, *even with stronger hardness assumptions than those in the classical setting*. This motivates our second question:

Question 2: Do there exist black-box, constant-round constructions of post-quantum MPC with ε-simulation?

We remark that the recent breakthrough by Lombardi, Ma, and Spooner [LMS22] proposed a new model for post-quantum simulation, called coherent-runtime expected quantum polynomial time simulation. In this model, a simulator is allowed to coherently run multiple computational branches with different runtime so that they can interfere with one another. They show a set of results in this model that bypass the impossibility result of [CCLY22b]. We emphasize that in the current work, we focus on the traditional notion of quantum *strict*, rather than *expected*, polynomial-time simulation. It is also worth mentioning that although the [LMS22]'s coherent-runtime expected QPT simulation implies ε-simulation, the round complexity of *fully* black-box PQ-MPC has not been resolved in their model either. We leave it as an interesting direction to investigate the implications of the [LMS22] model on the round complexity of black-box PQ-MPC.

1.1 Our Results

In this work, we give a positive resolution of these two questions.

1.1.1 Black-Box PQ-2PC and PQ-MPC with Full Simulation

We obtain the first fully black-box PQ-2PC protocol from minimal assumptions, in any super-constant number of rounds, which is (asymptotically) optimal for black-box simulation (due to the lower bound of [CCLY22b]):

Theorem 1. *There exists a $\omega(1)$-round,[7] black-box construction of PQ-2PC (with full simulation), from the minimal assumption of post-quantum, semi-honest oblivious transfers (OTs).*

To build this protocol, we follow the approach of [CCLY22a] originally designed for black-box PQ-2PC with *ε-simulation*. Very roughly speaking, the most crucial component in their approach is a *post-quantum extractable commitment* with ε-simulation. This primitive is similar to the standard notion of extractable commitments in the classical setting, but it additionally requires that the post-extraction state of C^* (the malicious committer) should be ε-indistinguishable from that in the real execution.[8] We observe that we can use [CCLY22a] template to also achieve the standard notion

[7] While the term $\omega(1)$ is typically used for lower bounds, in our context, we use it to mean that "any super-constant value suffices."

[8] We remark that while simulating for C^*'s post-extraction state is trivial in the classical setting, this task is particularly challenging when C^* is a quantum machine (see [CCLY22a]).

of fully simulatable PQ-2PC (instead of just ε-simulatability) if we can just make the underlying extractable commitment fully-simulatable.

While the goal is clear, achieving this turns out to be quite non-trivial. To the best of our knowledge, all existing black-box constructions for this task crucially utilize quantum communication in their protocol [BCKM21, GLSV21]. Since our aim is to build a *post-quantum* protocol, this does not suit us. To address this issue, we introduce the first black-box construction of post-quantum extractable commitments with full simulation. Our construction makes use of post-quantum semi-honest OTs. We note that while semi-honest OTs may not be the minimal assumption for extractable commitments per se, it is however minimal for our eventual goal of PQ-2PC.

Lemma 1. *Assuming the existence of post-quantum semi-honest OTs, there exists a $\omega(1)$-round, black-box construction of post-quantum extractable commitments with full simulation.*

Given our construction of black-box PQ-2PC, we can use it to get a construction for fully simulatable PQ-MPC. This is done by invoking the [IPS08] black-box compiler to get a polynomial round PQ-MPC — the key thing to notice is that our 2PC construction can also serve as the kind of OT protocol that is required by this compiler, albeit necessitating sequential composition for multiple OT calls. We refer the reader to Sec. 16 for further details.

Theorem 2. *There exists a black-box construction of PQ-MPC with full simulation, from the minimal assumption of post-quantum semi-honest OTs.*

1.1.2 Application I: LOCC MPC without OWFs

A recent breakthrough by Kretschmer, Qian, and Tal [KQT25] constructed a classical oracle relative to which $\mathbf{P} = \mathbf{NP}$, yet $\mathbf{BQP}$-computable (and quantum-secure) trapdoor OWFs exist, making them impossible to "de-quantize" in a black-box manner. This relativized world is particularly surprising when contrasted with its classical counterpart, where $\mathbf{BPP}$-computable OWFs can be de-randomized in a black-box manner [IL89].

[KQT25] established their main theorem via a fully black-box reduction. Consequently, relative to the same classical oracle, their theorem extends to demonstrate the existence of any "LOCC" cryptographic object that admits a fully black-box reduction to trapdoor OWFs in the post-quantum setting. Here, LOCC stands for "local operations and classical communication," meaning that parties can perform local quantum operations, but all communication must be classical.

By combining our Thm. 1 (and Thm. 2) above *and* the post-quantum fully black-box reduction from semi-honest OTs to trapdoor OWFs from [GKM+00][9], the authors of [KQT25] were able to derive the following Corollary 1 as a corollary of their main theorem. As explained in [KQT25], our Thm. 1 (and Thm. 2) are essential for this result, as previous 2PC/MPC constructions either make non-black-box use of semi-honest OTs or lack security proofs in the presence of a quantum attacker.

Corollary 1 ([KQT25, Corollary 39], strengthened[10]). *There exists a classical oracle relative to which classical-communication and quantum-secure MPC exist, yet $\mathbf{P} = \mathbf{NP}$.*

[9] Although the original work [GKM+00] was focused on the classical setting, it is straightforward to see that their reduction holds in the post-quantum setting as well.

[10] The original [KQT25, Corollary 39] relied only on our Thm. 1 to obtain maliciously secure OTs (and thus 2PC). Here, we extend it to MPC using the stronger Thm. 2.

1.1.3 Constant-Round Black-Box PQ-MPC with ε-Simulation

As for ε-simulation, we study the general multi-party setting, and obtain the first constant-round fully black-box construction for PQ-MPC by relying on the same (more accurately, the post-quantum analog of) hardness assumptions as for the state-of-the-art *classical* MPC protocols:

Theorem 3. *There exists a constant-round black-box construction of ε-simulatable PQ-MPC from a variety of standard post-quantum cryptographic primitives, such as lossy public-key encryption, linearly homomorphic public-key encryption, or dense cryptosystems.*[11]

Our approach to Thm. 3 follows a pipeline established for classical constant-round black-box MPC, which has evolved through a series of prior work [IPS08, PW09, Wee10, Goy11, GLOV12]. In broad terms, we demonstrate that if the building components used in this pipeline are properly instantiated using their post-quantum equivalents, the outcome can be extended to the post-quantum realm. Further insights into this process are elaborated upon in Sec. 2.1. For now, it is worth noting that a critical step in this framework is the development of a black-box *1-many non-malleable* commitment scheme in constant rounds. This constitutes the primary technical challenge in the post-quantum setting.

Post-Quantum 1-Many Non-Malleability. Non-malleable commitments [DDN91] are commitments secure in the so-called *man-in-the-middle* (MIM) setting: An adversary $\mathcal{M}$ plays the role of a receiver in one instance of a commitment (referred to as the *left session*), while simultaneously acting as a committer in another session (referred to as the *right session*). During the execution, $\mathcal{M}$ can potentially make the value committed in the right session depend on that in the left session, in a malicious manner that is to her advantage. Notice that this is not breaking the hiding property of the commitment scheme, as $\mathcal{M}$ may be able to conduct the above attack without explicitly learning the value committed in the left session. Furthermore, a commitment is said to be *1-many* non-malleable if it is secure in the MIM setting with one left session but *polynomially many* right sessions, i.e., the adversary $\mathcal{M}$ cannot make the *joint distribution* of the values committed across all right sessions depend on the one committed in the left session.

In the classical setting, the existence of black-box constant-round 1-many non-malleable commitments was established under the minimal assumption of one-way functions [Goy11, GLOV12]. Such commitments played a pivotal role in enabling black-box constant-round MPC. However, in the post-quantum context, achieving non-malleability (even in the 1-1 MIM setting) with constant rounds proves to be an exceptionally challenging task. A recent result by [LPY23a] succeeded in obtaining a post-quantum 1-1 non-malleable commitment in constant rounds. Yet, their construction relies significantly on *non-black-box* usage of post-quantum one-way functions, and it remains uncertain if their scheme can maintain non-malleability in the more demanding 1-many scenario.

In this work, we obtain a black-box and constant-round construction for a *weak version* (explained shortly) of 1-many post-quantum non-malleable commitments, from the minimal assumption of post-quantum one-way functions. Compared to the standard notion of 1-many non-malleability, our construction is restricted in the following sense:

– It supports a polynomial *tag space*[12], instead of a exponential-size tag space as required by the standard definition.

[11] We did not mention post-quantum (enhanced) trapdoor permutations as they are not known from standard quantum hardness assumptions yet. But as long as they exist, they can be included in Thm. 3 as well.

[12] Each execution of non-malleable commitments requires a unique tag; otherwise, it is impossible to protect against MIM attacks (see [Pas04] for related discussions).

– It is non-malleable only in the *synchronous* setting, meaning that all the messages of the left session and the polynomially many right sessions are sent in parallel.

– It is non-malleable conditioned on the fact that the honest receiver *in every right session* accepts. That is, if there is some right session where the receiver rejects during the commit stage (the committed value for this session is then defined to be $\perp$), then our protocol does not provide any non-malleability guarantee. (We refer to Def. 6 for a formal treatment.)

We emphasize that while our construction may not be as powerful as the standard 1-many post-quantum non-malleable commitments, it already has non-trivial applications. Firstly, such a scheme suffices for our main focus of post-quantum MPC. Additionally, it also reduces the assumption utilized in a lower bound of quantum parallel repetition (as we will discuss shortly). We believe it will find more applications in the future.

Theorem 4. *Assuming the existence of post-quantum one-way functions, there exists a black-box and constant-round construction of* weak *(as explained above) post-quantum 1-many non-malleable commitments.*

It is known that 1-many non-malleability implies the seemingly more demanding *many-many* non-malleability, using a standard hybrid argument. This reduction holds even in the post-quantum setting (see e.g., [ABG+21, Lemma 7.3]). This yields the following corollary of Thm. 4.

Corollary 2. *Assuming the existence of post-quantum one-way functions, there exists a black-box and constant-round construction of* weak *(as explained above) post-quantum many-many non-malleable commitments.*

1.1.4 Application II: Quantum Parallel Repetition Lower Bound

Interestingly, our many-many non-malleability commitments find further application in establishing the lower bound for parallel repetition of post-quantum arguments. The recent work by Bostanci, Qian, Spooner, and Yuen [BQSY24] shows that parallel repetition does not always reduce the soundness error of post-quantum interactive argument systems. In particular, for any polynomial $k(\lambda)$, the authors of [BQSY24] constructed a *constant-round* interactive argument for which a k-fold parallel repetition does not reduce the (post-quantum) soundness at all. Their construction makes use of many-many post-quantum (synchronous) non-malleable commitments in constant rounds, which were not known previously. Now, the above Corollary 2 reduces the assumption used in [BQSY24] to the existence of post-quantum one-way functions. We state the result in the following Corollary 3 and refer the interested reader to [BQSY24, Theorem 1.6 and Section 6][13] for more information.

Corollary 3. *Assume the existence of post-quantum one-way functions. Then, for every polynomial $k(\lambda)$, there is a constant-round post-quantum interactive argument such that a $k(\lambda)$-fold repetition does not decrease the soundness error compared to the original protocol.*

1.2 More Related Work on Non-Black-Box Constructions

Besides the aforementioned works [ABG+21, LPY23a, GLM23], other non-black-box constructions of PQ-2PC also exist, such as [LN11, HSS11]. This naturally raises the question: how large is the

[13] We remark that [BQSY24, Theorem 1.6] assumes 'concurrent-secure' many-to-many non-malleable commitments. But as the authors have shown in [BQSY24, Section 6], 'parallel-secure' (i.e., synchronous) many-to-many non-malleable commitments suffice. Moreover, in their application, if the verifier in one session rejects, the entire execution is considered rejected. Thus, our weak many-many non-malleability notion suffices.

gap between these non-black-box PQ-2PC protocols and our black-box PQ-2PC in Thm. 1? What are the key obstacles preventing the removal of non-black-box components in these constructions?

In fact, these works adopt a fundamentally different approach from ours, as we elaborate below.

[LN11] primarily focused on feasibility results rather than the black-box nature of the construction. Indeed, it is unclear how to remove the non-black-box components from the [LN11] approach. This is because [LN11] builds PQ-2PC following the GMW approach [GMW87]: first constructing a semi-honest protocol and then achieving active security by adding ZK proofs on each message to enforce honest behavior from the parties. This GMW approach is inherently non-black-box due to its reliance on ZK proofs for cryptographic statements (i.e., the parties' next-message functions).

Even in the classical setting, black-box constructions of 2PC/MPC move away from the GMW approach and instead follow a very different path established by the line of works [IPS08, PW09, CDMW09, Hai08, IKLP06, Wee10, Goy11]. Briefly, the key advantage of this line of work lies in the development of techniques that enforce honest behavior without requiring ZK proofs for cryptographic statements, while achieving a constant number of interactions. Our constructions follow this line of work in the post-quantum setting, and therefore have little overlap with the [LN11] approach.

A similar situation applies to [HSS11]. Essentially, the PQ-2PC from [HSS11] follows the classical approach established by [CLOS02]. This is another inherently non-black-box approach where a commit-and-prove protocol is executed on cryptographic languages to enforce honest behavior from the parties. This can be viewed as a variant of the GMW compiler in the Universal-Composable (UC) framework. As such, there is little common ground for further comparison.

2 Technical Overview

This section provides an overview of our techniques. We first discuss our construction of ε-simulatable PQ-MPC (i.e., Thm. 3). This is covered in Sec. 2.1 to 2.6. After that, we describe our approach to PQ-2PC and PQ-MPC *with full simulation* (i.e., Thm. 1 and 2). This is covered in Sec. 2.7 and 2.8.

2.1 Reduction to Post-Quantum 1-Many Non-Malleability

As mentioned earlier, our approach to black-box ε-simulatable post-quantum MPC follows a pipeline established in the classical setting. In the following, we first recall it.

Classical Framework. In the classical setting, the aforementioned pipeline to obtain constant-round and black-box MPC proceeds as follows:

1. *Malicious-Sender OT:* First, build a 1-out-of-2 string OT with a weak property, namely, with security against malicious senders but only *semi-honest* receivers; Additionally, the associated simulator for proving security is required to be 'straight-line' (i.e., not performing any rewindings). Such schemes are known from any of the following: certifiable enhanced trapdoor permutations, dense cryptosystems, linearly homomorphic PKE, or lossy PKE (see, e.g., [CDMW09, Wee10]). These schemes are black-box constructions and constant-round (indeed, two rounds suffice).
2. *Multi-Party Parallel OT:* Next, a compiler is employed to transfer the malicious-sender OT to a fully-secure OT *in the n-party parallel setting*. This is the setting of n parties where every pair of parties (P_i, P_j) runs two executions of same OT protocol, one with P_i as the sender and the other with P_j as the sender. All of these $2 \cdot \binom{n}{2}$ executions happen in parallel. Such a compiler was constructed in [Wee10, Goy11], which is constant-round and makes only black-box use of its building blocks. (We provide more details when describing our approach.)

3. *General-Purpose MPC:* Finally, another black-box compiler is employed to transfer the n-party parallel OT to a general-purpose n-party secure computation protocol. This compiler was introduced in [IPS08]. It blows up the round complexity only by a constant number.

Our Approach. At a high-level, our approach is to replace all the primitives employed in the above pipeline with their post-quantum analog, preserving both the constant-round and black-box properties.

First, we notice that Step 1 extends to the post-quantum setting straightforwardly. That is, post-quantum malicious-sender OTs (with straight-line simulation) can be based on post-quantum dense cryptosystems, linearly homomorphic PKE, or lossy PKE, which can be in turn based on the quantum hardness of Learning with Errors (QLWE).

Obtaining the post-quantum analog of Step 2 represents the main technical challenge. Let us first discuss about Step 3, assuming the existence of post-quantum multi-party parallel OTs (with ε-simulation). For that purpose, we notice that the same [IPS08] compiler (introduced in the classical setting) can be used to convert any *post-quantum* multi-party parallel OT (with ε-simulation) to a *post-quantum* MPC (with ε-simulation)[14]. It adds at most constant rounds, makes only black-box use of the given OT protocol, and does not rely on any extra assumptions. Roughly, this is because the original security reduction in [IPS08] is in straight-line and does not copy (or 'clone') the state of the adversary. Thus, the same proof can be migrated to the post-quantum setting. Although there are some caveats (e.g., how to handle the ε-simulation error), we choose not to expand on them in this overview and refer the reader to Sec. 12 for more details.

In the following, we focus on the post-quantum analog of Step 2.

Post-Quantum Multi-Party Parallel OT. Our starting point is the constant-round, black-box compiler described in [Wee10, Goy11]. The specific structure of this protocol is not the primary emphasis of this overview and is therefore omitted (see Sec. 11 for details). Our sole concern lies in the fact that this compiler relies on a distinct commitment scheme that enjoys the following properties:

- *Constant-Round and Black-Box:* This is necessary because our ultimate goal is to obtain a constant-round OT (and MPC) that makes only black-box use of the building blocks.
- *Parallel-Extractable:* It considers the setting where a potentially malicious committer executes n sessions of the scheme in parallel. It requires the existence of an extractor that can extract the committed values *in all the n sessions simultaneously.*
- *1-Many Non-Malleable:* As explained in the introduction, this notion considers a MIM adversary $\mathcal{M}$ who plays the role of a receiver in one instance of the commitment (dubbed the left session), while simultaneously acting as a committer in polynomially many other instances (referred to as the right sessions). All the sessions happen in parallel. For security, we require that $\mathcal{M}$ cannot correlate the *joint distribution* of the values committed in all of the right sessions with that in the (single) left session. (See Sec. 3.4 for a formal definition.)

It can be shown that as long as we have a post-quantum analog of the above commitment scheme, the same [Wee10, Goy11] compiler can be used to convert a post-quantum malicious-sender OT (with ε-simulation) to a post-quantum multi-party parallel OT (with ε-simulation).

It is worth noting that the post-quantum equivalent of (parallel) extractability necessitates an additional requirement: the extractor must be capable of simulating the post-extraction state of the malicious committer. This aspect was not explicitly addressed in the classical setting because

[14] The same observation has been made in the two-party setting in [CCLY22a].

classical information can be 'cloned,' allowing an extractor to create two copies of the committer—one for extraction and the other for simulating the post-extraction state, thereby mimicking a straight-line execution. However, achieving such a 'simulatable' extraction becomes challenging in the post-quantum realm. In fact, there have been suggestions that achieving post-quantum extractable commitments with negligible simulation error may be impossible in constant rounds, if one insists on black-box simulation techniques [CCLY22b, CCLY22a, CCLL24]. This very challenge is the reason why we relax our security notion to ε-simulatability. Looking forward, our objective is to aim for parallel-extractable commitments with ε-simulation for post-extraction state in the post-quantum setting. We will demonstrate that this suffices for the [Wee10, Goy11] compiler when our ultimate goal is ε-simulatable PQ-MPC. To maintain our focus on the core topics of this overview, we will omit additional details in this regard and refer the reader to Sec. 12.

Post-Quantum 1-Many Non-Malleability. Next, our focus turns to the development of a commitment scheme that satisfies the post-quantum analog of the three properties mentioned earlier. To achieve this, we start with the constant-round post-quantum non-malleable commitment described in [LPY23a]. First, we observe that the [LPY23a] scheme is already post-quantum extractable (with ε-simulation of the post-extraction state) *in the stand-alone setting*. Also, it is not hard to see that the techniques from [CCLY22a] can be used to prove that the [LPY23a] scheme is post-quantum *parallel*-extractable as well. However, it is important to note that the [LPY23a] scheme achieves non-malleability in the *1-1 setting* only, as opposed to being *1-many non-malleable*. Furthermore, it extensively relies on the use of its underlying primitive (i.e., a post-quantum one-way function) in a *non-black-box* manner. Indeed, the question of constructing constant-round post-quantum commitments that achieve *either of these two properties* remains an open challenge. In the following, we describe our ideas to achieves *both* properties, under the minimal assumption of post-quantum one-way functions.

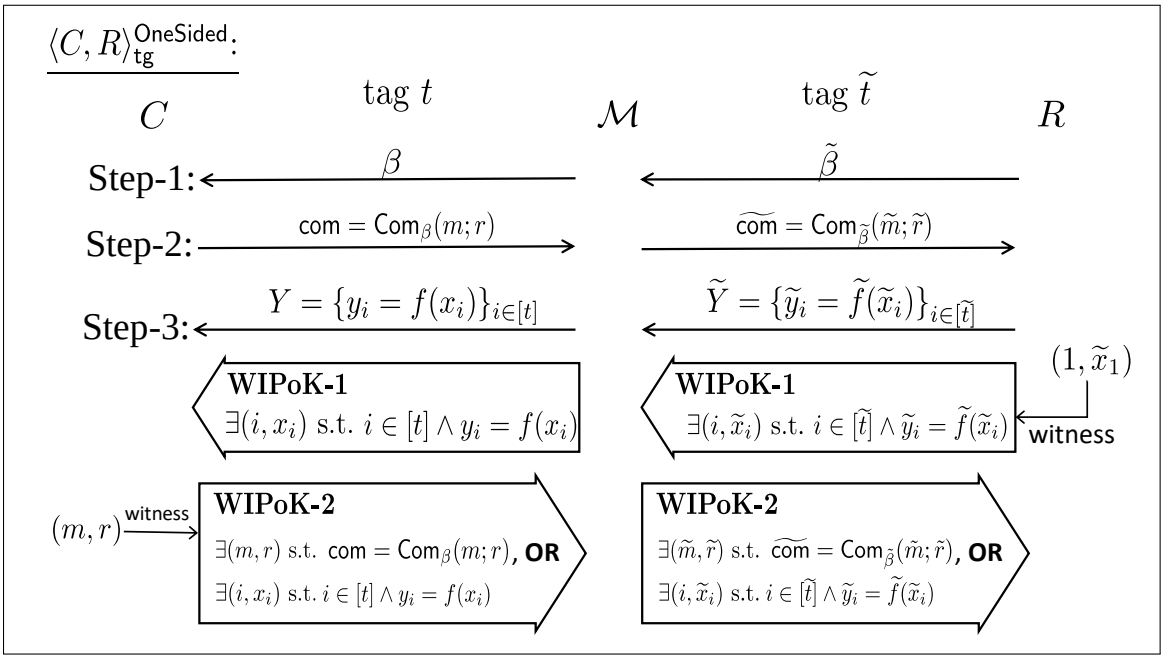
2.2 PQ-NMC from [LPY23a]

We first recall the [LPY23a] construction and the salient features therein that help with the proof of non-malleability. For our purpose, it is sufficient to focus on the simplified scheme shown in the technical overview of [LPY23a]. That construction achieves non-malleability in the synchronous 1-1 MIM setting, where the left-session tag t is *strictly smaller than* the right-session tag $\widetilde{t}$ (dubbed 'one-sided' non-malleability).

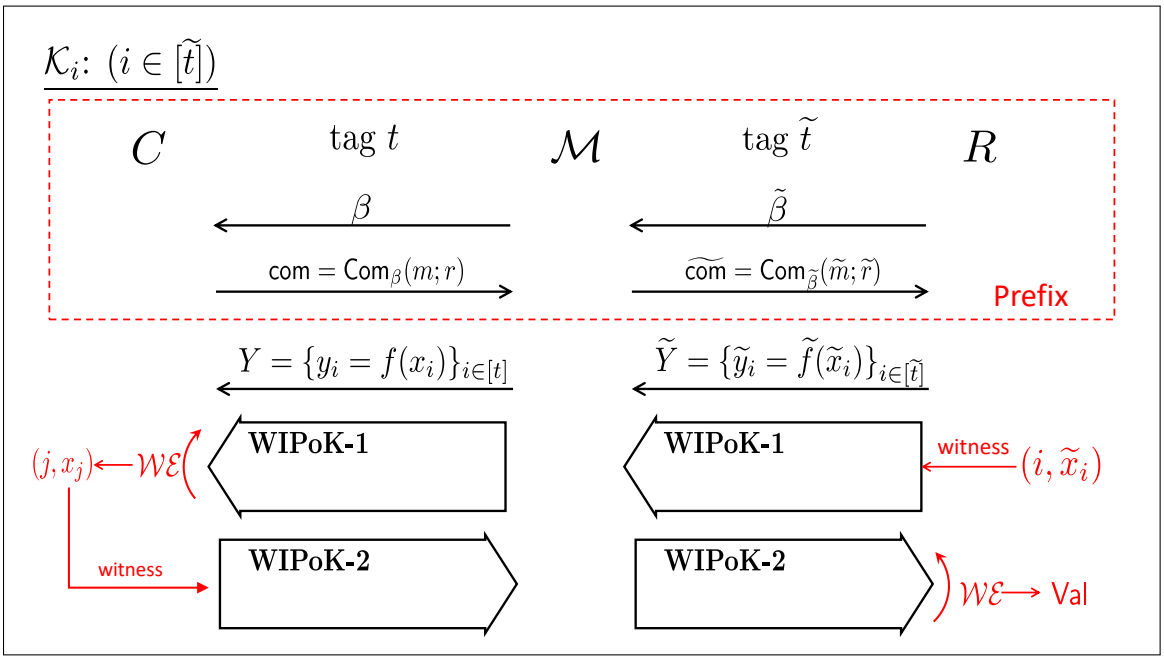
It works as follows: To commit to a message m with tag $t \in [n]$, the committer C first commits to m using a statistically binding commitment scheme $\mathsf{com} = \mathsf{Com}(m; r)$ (e.g., Naor's commitment). Then, the receiver R sends a hard puzzle that has *exactly* t distinct solutions; R also gives a witness-indistinguishable proof of knowledge (referred to as **WIPoK-1**) to prove that it knows one of the t solutions. Finally, C is required to prove using another WIPoK (referred to as **WIPoK-2**) that it knows *either* the value committed in com *or* one solution to R's hard puzzle.

We illustrate the 1-1 MIM execution of this protocol in Fig. 1a (borrowed from [LPY23a]), where the t-solution hard puzzle is instantiated with t images $(y_1, \ldots, y_t)$ of an *injective* OWF f, and the solutions are the preimage x_i's satisfying $y_i = f(x_i)$ for all $i \in [t]$.

A Pigeon-Hole Argument. Proofs of non-malleability typically rely on the following intuitive claim: in the MIM interaction, we want the honest committer C to be able to 'cheat' on the left, while the MIM adversary $\mathcal{M}$ should not be able to 'cheat' similarly on the right. To show this, [LPY23a] relies on a pigeon-hole based argument, which we sketch here. Note that in the MIM interaction depicted in Fig. 1a, there is an inherent asymmetry between the left and right side



(a) Man-in-the-Middle Execution of $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{OneSided}}$



(b) The Simulation-less Extractor $\mathcal{K}_i$

Fig. 1

executions—there are more puzzle solutions on the right as compared to the number on the left (since $t < \widetilde{t}$). This leads to the following intuitive observation: suppose the receiver R switches the witness it uses in **WIPoK-1** on the right. Due to this asymmetry, $\mathcal{M}$ cannot switch its witness in every such case. Namely, by the pigeonhole principle, there *must* exist indices $i, j \in [\widetilde{t}]$ on the right and $k \in [t]$ on the left such that no matter which of $\widetilde{x}_i$ or $\widetilde{x}_j$ is used as a witness, $\mathcal{M}$ can only x_k as witness in the left **WIPoK-1**.

For this 'pigeon-hole tuple' $(\widetilde{x}_i, \widetilde{x}_j, x_k)$, we see that the following must also happen—suppose the left-session C uses x_k as witness in the left **WIPoK-2**, we can then argue that $\mathcal{M}$ must use $\widetilde{m}$ in the right **WIPoK-2** as follows:

- First, assuming R uses $\widetilde{x}_i$ in the right **WIPoK-1** and $\mathcal{M}$ uses x_k in the left **WIPoK-1**, if we extract from the right **WIPoK-2**, the extracted value can only equal $\widetilde{m}$ or $\widetilde{x}_i$. That is because other $\widetilde{x}_j$'s (with $j \neq i$) have not been used in the right **WIPoK-1** and so we can appeal to the one-wayness of f to say that $\mathcal{M}$ cannot learn these values.
- Similarly, assuming R uses $\widetilde{x}_j$ in the right **WIPoK-1** and $\mathcal{M}$ uses x_k in the left **WIPoK-1**, if we extract from the right **WIPoK-2**, the extracted value can only take the values $\widetilde{m}$ or $\widetilde{x}_j$.

Then, by the witness indistinguishability of the right **WIPoK-1**, the extracted value should *not* change if R switches between $\widetilde{x}_i$ and $\widetilde{x}_j$ (in the right **WIPoK-1**). Thus the extracted value can only be $\widetilde{m}$ if R uses $\widetilde{x}_i$ (or $\widetilde{x}_j$) on the right and C uses x_k on the left.

It seems that the approach outlined above is promising and can help show the intuitive guarantee of allowing C to "cheat" on the left while preventing such behavior from $\mathcal{M}$ on the right. However, actually proving such a guarantee is quite challenging and is the core technical contribution of [LPY23a]. In particular, they must address the following technical hurdles: (1) a mechanism is needed to efficiently identify the 'magic' triples $(\widetilde{x}_i, \widetilde{x}_j, x_k)$—one cannot extract x_k from the left **WIPoK-1** simply by rewinding, because the above argument relies on the WI property of that stage, which may not hold if it is rewound. (2) More crucially, the above pigeon-hole argument assumed a one-to-one correspondence between the $\mathcal{M}$'s witness and R's witness used in **WIPoK-1**. This is over simplified. Indeed, $\mathcal{M}$ can switch its witness *probabilistically* when R switches witnesses.

To address these issues, [LPY23a] develops an involved *distributional* pigeon-hole lemma to formally captures the intuition above. For the current overview, the details of this lemma is not crucial, and thus we do not dig it further. However, the structure of the proof in [LPY23a] is crucial for understanding our new techniques later. Therefore, we briefly recall its structure below, focusing only on the aspects necessary to establish a foundation for the subsequent discussion of our techniques.

[LPY23a]'s Proof Structure. At a high level, the [LPY23a] approach involves a reduction from non-malleability to the hiding of the left Naor commitment Com performed initially. This is a rigorous formalization of the aforementioned intuition that 'we can cheat in the left but $\mathcal{M}$ cannot in the right.' In more detail, they first make the subsequent portion of the left execution after Com *independent* of message m, so that the reduction can go through. Next, the idea is to extract the $\widetilde{m}$ committed initially by $\mathcal{M}$ from the right session. If one can always extract the correct value $\widetilde{m}$ and while not unduly disturbing $\mathcal{M}$'s post-extraction state, the reduction to the hiding of Com is easily seen: If $\mathcal{M}$'s $\widetilde{m}$ changes according to m (i.e., the message committed in Com on the left), then one can always use $\widetilde{m}$ extracted from the right **WIPoK-2** to detect the difference, compromising the hiding property of Com. Thus, the *most challenging part* in this approach is to efficiently extract $\widetilde{m}$, without disturbing $\mathcal{M}$'s post-extraction state too much.

[LPY23a] builds such an 'extractor with simulation' in two steps. First, they build a 'base' extractor $\mathcal{K}$ without any simulation guarantee, whose job is only to extract $\widetilde{m}$ correctly. $\mathcal{K}$ works by sampling an uniform index $i \in [\widetilde{t}]$ and running the machine $\mathcal{K}_i$ depicted in Fig. 1b. In particular $\mathcal{K}_i$ differs from the real MIM execution in the following way: (1) it uses $\widetilde{x}_i$ (instead of $\widetilde{x}_1$) as the witness in the right **WIPoK-1**; (2) it uses the witness extractor $\mathcal{WE}$ to obtain the witness $j\|x_j$ used by $\mathcal{M}$ in the left **WIPoK-1**, and then uses this extracted $j\|x_j$ to finish the left **WIPoK-2**; (3) it uses the witness extractor $\mathcal{WE}$ to extract the witness Val used by $\mathcal{M}$ in the right **WIPoK-2** and hopes that $\mathsf{Val} = \widetilde{m}$.

[LPY23a] uses the aforementioned distributional pigeon-hole lemma to prove that such a machine $\mathcal{K}$ will indeed extract $\mathsf{Val} = \widetilde{m}$ with noticeable probability, *conditioned on a 'good' prefix (i.e., Steps 1 and 2) from which $\mathcal{M}$ will indeed finish the execution with noticeable probability* (this condition is necessary: one cannot hope to extract $\widetilde{m}$ with noticeable probability if, say, $\mathcal{M}$ always aborts in the real MIM execution).

Next, [LPY23a] develops a *simulation-extraction* lemma. Using this lemma, they are able to convert the simulation-less K into a new machine $\mathcal{SE}$ that extracts $\widetilde{m}$ while *also* being able to simulate the post-extraction state. As remarked before, this gets them most of the way through the proof—[LPY23a] show that one can use $\mathcal{SE}$ to complete the outlined reduction to hiding of Com, and thus demonstrate non-malleability.

2.3 Our Black-Box Construction: 1-1 Setting

While the [LPY23a] commitment is non-malleable and works in constant rounds, it does not suffice for our application because the construction makes heavy non-black-box use of its cryptographic components, and it is unclear if their security proof holds in the more demanding 1-many MIM setting. Now, we first introduce new ideas to obtain a black-box construction.

Observe that there are two sources of non-black-box usage in Fig. 1a. First, R's proof in **WIPoK-1** needs the code of the hard puzzle (i.e., the OWF f); Second, C's consistency proof **WIPoK-2** makes non-black-box use of *both* the hard puzzle *and* Naor's commitment in **Step-1**.

We first notice that it is not hard to make R's behavior black-box. Essentially, what R does in the hard-puzzle set-up stage is first 'committing' to t solutions and then proving that it knows one solution. This is actually a classical task called *witness indistinguishable commit-and-prove (of knowledge)*. It is not hard to modify existing black-box witness indistinguishable commit-and-prove protocols (e.g., [CCLY22a]) to make R's hard puzzle and the **WIPoK-1** steps black-box. In the following, we only focus on the non-black-box usage on C's side.

Making C's consistency proof **WIPoK-2** black-box turns out to be quite challenging. One may hope to re-use the aforementioned black-box commit-and-prove technique to resolve the non-black-box use of the **Step-1** Naor's commitment. However, the real difficulty lies in its dependency on the puzzle (i.e., the alternate clause in **WIPoK-2** in Fig. 1a). Notice that the puzzle solutions are *only known to R*! This means that the statement becomes one about the *preimages* of f, for which the committer/prover (i.e., party C) does not have a witness. In this scenario, it is unclear how the black-box commit-and-prove techniques could help.

We develop new ideas to tackle this challenge. Our guiding principle is to modify **WIPoK-2** so that it proves only *non-cryptographic* relations. That is, we try to make **WIPoK-2** depend only on values that are *either* committed by C itself (so that it can be handled by black-box commit-and-prove), *or* otherwise made available to both parties in the course of the protocol.

To do this, we start with careful scrutiny of [LPY23a]'s simulation-less extractor $\mathcal{K}_i$ shown in Fig. 1b. We observe that the **WIPoK-2** stage there can be interpreted as serving a *dual* function in their security proof: (1) It is used to ensure honest behavior of the committer (or $\mathcal{M}$ in the MIM

setting); This is what essentially helps them perform the distributional pigeon hole argument. (2) The proof of knowledge property of this stage provides *extraction opportunities* to efficiently learn $\widetilde{m}$.

We find that these two purposes can in fact be 'decoupled', leading to a more modular security proof as follows: first, one can imagine a $\mathcal{K}_i'$ that is identical to the original $\mathcal{K}_i$ but does not invoke $\mathcal{WE}$ for the right **WIPoK-2**. For this new $\mathcal{K}_i'$, we could hope to re-use [LPY23a]'s distributional pigeon-hole lemma to argue that *the witness 'used' by $\mathcal{K}_i'$ in the right* **WIPoK-2** *should be $\widetilde{m}$ with good probability.* Second, it is a simple application of the proof of knowledge property to extract the witness used in this **WIPoK-2**, *which is guaranteed to be $\widetilde{m}$ (with good probability) by the previous step.*

On the other hand, notice that *once the protocol is completed*, it is safe for R to disclose its hard puzzle solutions. This seems vacuously true and useless. However, it becomes very useful once combined with the observation in the last paragraph, which yields our key idea—we propose to replace the **WIPoK-2** stage by the following three steps:

1. **ExtCom:** C commits to m again using an extractable commitment;
2. **Solution Reveal:** R reveals all the hard puzzle solutions $(x_1, \ldots, x_t)$;
3. **WIP-2:** C proves the same relation as in [LPY23a] **WIPoK-2**, using a WI proof. (Note that we do not require the proof of knowledge property anymore.) In particular, C proves that the value committed in ExtCom is equal to *either* the value committed in **Step-1** Naor's commitment, *or* one of the puzzle solutions among the (already revealed) x_i's.

This structure exercises our previous observations as follows. We first 'decouple' the two functions of the original **WIPoK-2** (as explained above) by delegating the extractability to **ExtCom** and the consistency proof to **WIP-2**. Then, we can ask R to reveal the puzzle solutions right after **ExtCom**, because at that moment C (or $\mathcal{M}$ in the MIM setting) has already 'fixed' the witness for consistency proof (i.e., the current **WIP-2**) in **ExtCom**, and cannot change its mind anymore even if the hard puzzle solutions are revealed to it.

Indeed, we can show this protocol is non-malleable as follows. Consider a $\mathcal{K}_i''$ that is similar to $\mathcal{K}$ in Fig. 1b, but instead extract Val from $\mathcal{M}$'s **ExtCom** on the right. Then, a similar distributional pigeon-hole argument can be established, proving that the Val extracted from **ExtCom** indeed equals to $\widetilde{m}$ with good probability. Then, using the same technique as in [LPY23a], we can build a simulation-less extractor $\mathcal{K}$ from this new $\mathcal{K}_i''$ and convert $\mathcal{K}$ to a simulation extractor to finish the final reduction to the hiding property of the left Naor's commitment.

It seems we are already done—To make C's behavior black-box, note that all C does now is to commit to two values, one in the original **Step-1** and the other in the new **ExtCom**, and then proves in the new **WIP-2** a non-cryptographic predicate (since the puzzle solutions are revealed) over the two committed values and the revealed puzzle solutions. As mentioned earlier, this task can be made black-box by a simple application of known black-box commit-and-prove techniques. This indeed works *if our goal were to build a classically secure scheme.* Unfortunately, this step turns out to be challenging in the presence of a quantum $\mathcal{M}$, due to reasons exclusive to the quantum setting. We describe these in Sec. 2.4.

2.4 Quantum-Exclusive Challenges

To explain these challenges, we first briefly recall how canonical black-box commit-and-prove protocols broadly work. At a very cursory level, such protocols have (as indicated by their nomenclature) well-defined *commit* and *prove* stages. The **commit stage** has the prover commit to values involved

later in the proof statement, but not directly. Instead, one commits to shares obtained from a *Verifiable Secret Sharing* (VSS) of the intended value. These are a strengthening of standard secret secret sharing schemes and allow for reconstruction of the secret even if some shares are adversarially tampered with—and like standard secret sharing schemes, hide the secret completely if not enough shares are collected.

This is done for compatibility with the subsequent (black-box) **prove stage**, where the prover follows the *MPC-in-the-head* [IKOS07, GLOV12] approach: First the committer (or prover) C emulates in her head a MPC execution with n parties (where the inputs are the VSS shares from the commit stage), and commits to the views of each party during this virtual execution. This is followed by a 'cut-and-choose' interaction, where C and the receiver (or verifier) R agree on some subset $\eta \subset [n]$ (of size k) of the views from the virtual MPC execution (which includes the initial VSS shares)—typically via a coin-tossing step which helps with extablishing zero knowledge for this protocol, but we gloss over this for now—and the prover reveals the corresponding views. The verifier then checks *consistency* of the views; by the design of the protocol, this allows the verifier to catch a cheating prover out with a fairly high probability (establishing soundness), but reveals nothing about the value committed by the prover in the commit stage (leading to zero knowledge).

As indicated, we want to use such a scheme to supply the commmitter's proof of consistency. Accordingly, we will have **Step-1** and the new **ExtCom** correspond to the commit stage, and the **WIP-2** will correspond to the prove stage of the commit-and-prove protocol. In particular, this makes the initial commitment in **Step-1** of our protocol no longer a straightforward Naor commitment to m—instead, this is now a commitment (in parallel) to *VSS shares* of m, as is necessary for the commit-and-prove technique. Since we aim to overall reduce non-malleability to the hiding of the initial commitment, we must now consider a reduction directly to the hiding of the commit stage of the commit-and-prove protocol.

This is deceptively tricky. The reason lies in the operation of the black-box commit-and-prove protocol; recall that this entails that some of the VSS shares from the **commit phase** be revealed during the **prove stage**. To realize this with standard commitments is hard because the hiding guarantee does not cover such 'partial' revealing of information about the committed value. The solution turns out to involve modifying the standard hiding game to incorporate VSS shares (namely, to perform the commitment as described earlier by committing to VSS shares of the message) and allowing for a partial reveal of a certain subset shares of the adversary's choice later in the challenge (this 'mimics' the subsequent proof stage interaction, allowing for a subset of shares corresponding to η to be revealed by the VSS-based hiding challenger). We articulate such a 'VSS-based hiding' game in the course of our proof and show that this is hard to win given standard (computational) hiding of commitments and the secrecy property of VSS schemes.

We now turn to how the reduction itself works. As a first attempt, consider a reduction R^* that runs the MIM game internally and forwards the **Step-1** commitment to the external VSS hiding challenger. Note that R^* needs to specify a challenge set η. R^* could simply wait until the MIM $\mathcal{M}$ sends η in the left **WIP-2** execution, and forward this η to the external challenger. This seems to work, assuming R^* ran machine $\mathcal{K}_i''$ (or the $\mathcal{K}$ built out of $\mathcal{K}_i''$) to thereby win the VSS hiding game. However, note that in the final reduction to hiding, R^* instead needs to run the simulation extractor $\mathcal{SE}$ that is built out of $\mathcal{K}$ using the [LPY23a] simulation-extraction lemma. In more detail, $\mathcal{SE}$ involves *coherently rewinding* the machine $\mathcal{K}$. This is problematic: recall that $\mathcal{M}$ sends set η in

WIP-2, and this η needs to be forwarded to the external VSS hiding challenger, which we can of course not rewind [15].

Our heuristic towards solving this is to make sure that R^*'s communication with the external challenger should *end before the post-prefix phase of its internal MIM execution starts*. As a first try, we let R^* sample the set η by itself. To make sure that $\mathcal{M}$ indeed uses the same η during the left **WIP-2** stage, we add a coin-flipping protocol to generate the challenge set. The hope is to let R^* use the simulator for the coin-flipping interaction to 'force' $\mathcal{M}$'s challenge set to the η it sampled beforehand.

Unfortunately this does not work. Recall that we are in the MIM setting: if we force $\mathcal{M}$'s coin-flip protocol on the left, it is possible that $\mathcal{M}$ could in turn force the coin-flipping result in the right session with R, where the soundness guarantee of the right **WIP-2** may not hold anymore. It thus seems what we need is a *non-malleable* coin-flipping protocol; where we can force the result in the left session but $\mathcal{M}$ cannot. This however puts the cart before the horse in that it is known that *non-malleable* coin-flipping implies non-malleable commitments, which is our object. Thus it seems this approach is a dead end.

Trapdoor Coin-Flipping. Fortunately, we manage to resuscitate this approach with the following new ideas. To understand that, let us first delve deeper into the [LPY23a] proof of non-malleability. We have already mentioned that for the machine $\mathcal{K}_i''$ described before, even if we *do not* extract from the right **ExtCom**, we can use a similar pigeon-hole argument to show that the value committed there is indeed $\widetilde{m}$ with good probability. This is argued in two steps:

- First, we can use the [LPY23a] pigeon-hole argument to show that even if the left C commits in the left **ExtCom** to some $j\|x_j$ it extracted from the hard puzzle, $\mathcal{M}$ cannot in turn commit to some puzzle solution in the right **ExtCom**. We emphasize that this step *does not* rely on the soundness of the right **WIP-2**. (This is a feature inherited from [LPY23a] design, though there this observation was superfluous given that there was no call to 'decouple' the **WIPoK-2** into an **ExtCom** and a **WIP-2** as we do).
- Second, we can now invoke the soundness of the left **WIP-2**—since $\widetilde{m}$ and the right puzzle solutions are the only witnesses for the relation **WIP-2** proves, if the committed value in the right ExtCom is not any of the puzzle solutions (as argued above), then it must be $\widetilde{m}$.

It follows that even if we were to remove the **WIP-2** step in our protocol, it would still enjoy a limited form of 'non-malleability'—$\mathcal{M}$ cannot commit to a left hard puzzle solution *even if* the left C does so. Our idea is to leverage this limited 'non-malleability' by constructing a *limited* version of non-malleable coin-flipping that suffices for our purposes. Specifically, we construct what we call a *trapdoor coin-flipping* protocol. This is an augmented coin-flipping protocol between two parties C and R where C additionally commits to some string x before coin-flipping starts. The security guarantee of the actual coin-flipping stage is 'controlled' by the committed string x and a predicate $\phi(\cdot)$ given to both parties at the beginning of the coin-flipping stage. In particular, if the committed x satisfies the predicate $\phi(\cdot)$, then by design a committer can 'force' the coin-flipping result to a pre-sampled random string η; but if the committed x does not satisfy the predicate, then no QPT C^* can bias the coin-flipping result. (see Sec. 4 for a formal treatment).

We will use this trapdoor coin-flipping to determine the challenge in **WIP-2**, setting the trapdoor predicate to '**ExtCom** commits to one of the puzzle solutions.' Then for machine $\mathcal{K}_i''$, we can enforce the **WIP-2** challenge to η in the left execution using the trapdoor predicate (because C

[15] This is of course not an issue in the classical setting, wherein the machine $\mathcal{K}$ does not need enhancement, because simulation can be easily added on by making two copies of the adversary: one for extraction and the other for simulation.

does commit to some $j\|x_j$ in the left **ExtCom**). Then, we use the aforementioned 'limited non-malleability' guarantee to argue that $\mathcal{M}$ cannot force the coin-flipping for the right **WIP-2**, and thus the soundness guarantee of the right **WIP-2** still holds. This obviates the issues raised above.

Putting everything together, our proof works as follows:

- We first use the [LPY23a] pigeon-hole argument to argue that $\mathcal{M}$ cannot commit to any puzzle solutions in the right **ExtCom**. As mentioned above, this step does not make use of the soundness of **WIP-2**.
- Since the right **ExtCom** does not commit to any puzzle solution, the soundness of the right **WIP-2** must hold, *even if we enforce the coin-flipping step in the left* **WIP-2**. This comes as a guarantee of our trapdoor coin-flipping protocol. Therefore, by the soundness of the right **WIP-2**, the value committed in the right **ExtCom** must be $\widetilde{m}$ (and will be extracted by $\mathcal{K}''_i$).

As mentioned before, since we pre-sample η, R^*'s communication with the external VSS hiding challenger can be pushed entirely to the prefix phase. So R^* can make use of a 'full-fledged' simulation-extractor $\mathcal{SE}$ which is built from the new $\mathcal{K}''_i$ (indeed, from $\mathcal{K}$ that picks a random i and runs $\mathcal{K}''_i$ as in [LPY23a]), to complete the reduction.

Noisy Simulation-Extraction Lemma. We emphasize that the overview above forms only an intuitive explanation of our ideas. To implement them formally is more challenging as the aforementioned issues appear in a more subtle and technical manner. Owing to the paucity of space, we refer the reader to Sec. 5 for fuller details. However, there is a particularly subtle issue unique to our protocol (i.e., not appearing in [LPY23a]) that we would like to highlight. The above discussion pays much attention to the value committed in ExtCom. In certain steps of our proof, it becomes important to extract this value *efficiently*, in order to reduce the security to some *falsifiable*[16] assumptions. For that we often need to consider the extracted value and take it to be the committed message if it is *not* any of the puzzle solutions on the right. As reasoned above, this is the case in all but a noticeable fraction of cases.

However, this indeed starts affecting the conversion from $\mathcal{K}$ to $\mathcal{SE}$: the simulation-extraction lemma given in [LPY23a] crucially relies on the fact that the simulation-less extractor $\mathcal{K}$ will, if it does not abort, extract a *unique* string with good (technically, *noticeable*) probability. Put another way, $\mathcal{K}$ needs to know that what it extracts is indeed $\widetilde{m}$, and if not, it needs to output $\perp$. However, our new $\mathcal{K}$ cannot perform such checks—we only argued that $\mathcal{K}$ extracts $\widetilde{m}$ with noticeable probability. However, it could still be the case with noticeable probability that the extracted value is simply garbage or 'noise.' Even worse, $\mathcal{K}$ cannot detect this case because the **Step-1** commitment is performed in a black-box commit-and-prove format.[17] Fortunately, we can upgrade the simulation-extraction lemma from [LPY23a] to tolerate such noise, which suffices for our purpose. We refer to Sec. 7 for details.

This finishes the description of our 1-1 non-malleable commitments. The above discussion is based on the 'one-sided' (i.e., $t < \widetilde{t}$) scheme in the technical overview of [LPY23a] and thus our protocol inherits this restriction. Fortunately, we can remove this restriction using exactly the same 'two-slot' trick (initiated by Pass and Rosen [PR05]) as in [LPY23a]. We refer to Sec. 8 for details.

[16] An assumption is falsifiable [Nao03, GW11] if it can be modeled as an interactive game between an *efficient* challenger and an adversary, at the conclusion of which the challenger can *efficiently* decide whether the adversary won the game.

[17] This issue does not happen in [LPY23a] because their **Step-1** is Naor's commitment and $\mathcal{K}$ extracts the committed value together with the randomness from **WIPok-2**, so that it can check validity using Naor's decommitment algorithm.

2.5 One-Many Non-Malleability

So far, we have restricted our attention to the 1-1 setting for non-malleability. We now discuss our approach to obtain 1-many non-malleable commitments. Recall that this involves a man-in-the-middle adversary $\mathcal{M}$ that runs a single commitment session with an honest committer C on the left, and runs up to polynomially many commitment sessions on the right with honest receivers $R^{(1)}, \ldots, R^{(N)}$ (where we have used N to denote the total number of right sessions). We stress that 1-1 non-malleability does not directly imply 1-many non-malleability; this is a known barrier even in the classical setting and we elide further explanation for the sake of conciseness.

In spite of this, we are able to demonstrate that the same black-box construction we described above also enjoys 1-many non-malleability. In fact, we rely for the most part on very similar strategies to those we employed in the 1-1 case. The key similarity we exploit is that in the reduction to hiding that we outlined in the 1-1 case (that formed the base of our proof), the majority of the modifications we make are with respect to the *left* session—the only changes made on the right are changing which puzzle solution is committed to in **ExtCom** by the honest receiver, and *extracting* the committed value. We can thus hope to translate our technique to the 1-many setting as well. At a high level, we accomplish this in three steps:

Step 1. First, we design analogs of the simulation-less extractor $\mathcal{K}$ from earlier, that we call *localized* simulation-less extractors $\mathcal{K}^{(j)}$ (for $j \in [N]$). $\mathcal{K}^{(j)}$ performs extraction only in the j-th right session, and acts as an honest receiver in all other right sessions. It is not hard to establish the same guarantees for *each* $\mathcal{K}^{(j)}$ from the guarantees for $\mathcal{K}$ in the 1-1 setting, by means of a 'wrapper' reduction: namely, since the other right sessions of $\mathcal{M}$ are run honestly, we can run the 1-many interaction internally while treating the j-th session as the sole right session in a 1-1 MIM interaction, and handling all the other internal right sessions by itself.

Step 2. Next, we show how to build a *simultaneous* simulation-less extractor that is then able to extract the committed values in *all* right sessions with high success (but without any simulation guarantees). This step turns out to be particularly challenging due to the quantum nature of the MIM adversary $\mathcal{M}$. Note that the instanced extractors $\mathcal{K}^{(j)}$'s described above enjoys extractability only for a single run. If we want to perform simultaneous extraction, the natural attempt of running them one-by-one does not work—after the execution of, say, $\mathcal{K}^{(1)}$, the internal quantum state of $\mathcal{M}$ may have already been disturbed too much to support the execution of $\mathcal{K}^{(2)}$.

Another natural idea is to first convert $\mathcal{K}^{(1)}$ to a simulatable extractor, using the above noisy simulation-extraction lemma, and hope that the simulation guarantees of that lemma can 'protect' the state of $\mathcal{M}$ after the execution of $\mathcal{K}^{(1)}$ so that we can keep running $\mathcal{K}^{(2)}$ (and also convert $\mathcal{K}^{(2)}$ to a simulatable extractror to support $\mathcal{K}^{(3)}$ and so on). Unfortunately, this idea does not work either due to a subtle technical reason: The simulation guarantees of the noisy simulation-extraction lemma is for the *post-extraction* extraction state of $\mathcal{M}$, which means it simulates $\mathcal{M}$'s state at the end (or 'bottom') of the protocol; However, to be able to run $\mathcal{K}^{(2)}$ right after $\mathcal{K}^{(1)}$, we have to simulates $\mathcal{M}$'s state at the beginning (or 'top') of the protocol! Thus, new ideas are needed to resolve this problem.

We tackle this problem by crafting a novel *simultaneous* post-quantum extraction lemma, drawing upon the measure-and-repair technique introduced in [Zha20, CMSZ21]. This represents another main technical contribution of this work and it may find future applications where simultaneous post-quantum extraction is needed. We provide an overview of it in Sec. 2.6.

Step 3. Finally, we show that the same noisy simulation-extraction lemma described above can be used to upgrade this simultaneous simulation-less extractor to one *with simulation*, and thus finish the proof of non-malleability with a similar reduction to the VSS hiding game as in the 1-1 case.

2.6 Simultaneous Extraction

We now provide a brief overview of our simultaneous extraction lemma as mentioned above. We will use slightly different notation. In the following description, machines $\mathcal{K}_1, \ldots, \mathcal{K}_n$ play the role of our instanced simulation-less extractor $\mathcal{K}^{(1)}, \ldots, \mathcal{K}^{(N)}$ mentioned above (setting $n = N$). And $\mathcal{V}$ is a machine that should be treated as enforcing the condition that we start from a 'good' prefix (i.e., the **Step-1** Naor's commitment in the VSS form). It is necessary because as we mentioned earlier, we cannot hope to extract $\widetilde{m}$'s with good probability if $\mathcal{M}$ always aborts the execution before it naturally ends. The machine $\mathcal{K}$ is our desired simultaneous simulation-less extractor.

Lemma 2 (Simultaneous extraction lemma (informal)). *Let $\mathcal{V}$ and $\mathcal{K}_1, \ldots, \mathcal{K}_n$ be QPT algorithms for a polynomial n that satisfy the following:*

- **$\mathcal{V}$'s syntax:** *$\mathcal{V}$ takes a quantum state in Hilbert space $\mathcal{H}$ and outputs $\top$ or $\bot$;*
- **$\mathcal{K}_i$'s syntax:** *$\mathcal{K}_i$ takes a quantum state in Hilbert space $\mathcal{H}$ and outputs a classical string s_i or $\bot$;*
- **Uniqueness of $\mathcal{K}_i$'s output:** *For each $i \in [n]$, there is a classical string s_i^* such that $\mathcal{K}_i$'s output is either s_i^* or $\bot$ on any input;*[18]
- **"Good" states for $\mathcal{V}$ is also "good" for $\mathcal{K}_i$:** *For any noticeable γ, there is noticeable δ such that for any quantum state ρ, if*
$$\Pr[\mathcal{V}(\rho) = \top] \geq \gamma,$$
then
$$\Pr[\mathcal{K}_i(\rho) = s_i^*] \geq \delta.$$

Then there is a QPT algorithm $\mathcal{K}$ (called a simultaneous extractor) satisfying the following:

- **$\mathcal{K}$'s syntax:** *$\mathcal{K}$ takes a quantum state in Hilbert space $\mathcal{H}$ and outputs n classical strings $(s_1, s_2, \ldots, s_n)$ or $\bot$;*
- **Uniqueness of $\mathcal{K}$'s output:** *$\mathcal{K}$'s output is either $(s_1^*, s_2^*, \ldots, s_n^*)$ or $\bot$ on any input;*
- **"Good" states for $\mathcal{V}$ is also "good" for $\mathcal{K}$:** *For any noticeable γ, there is noticeable δ' such that for any quantum state ρ, if*
$$\Pr[\mathcal{V}(\rho) = \top] \geq 8\gamma,$$[19]
then
$$\Pr[\mathcal{K}(\rho) = (s_1^*, s_2^*, \ldots, s_n^*)] \geq \delta'.$$

If the input ρ is classical, then the above lemma trivially holds: $\mathcal{K}$ can simply run each $\mathcal{K}_i$ many times until it succeeds. However, if ρ is quantum, the state may collapse once we run $\mathcal{K}_i$ for some i, which prevents us from running it on the same state again. To resolve the issue, our idea is to use the "state repairing" technique introduced in [CMSZ21]. To explain their technique, we first review the concept of (approximate) projective implementation introduced by Zhandry

[18] The formal version of this lemma (see Lem. 18) permits the outputs of $\mathcal{K}_i$ to be other 'noise' values, provided that the probability of this occurrence can be bounded by a noticeable function. This is essential for compatibility with the previously described noisy simulation-extraction lemma. However, for this overview, we overlook this detail to maintain focus on the main idea.

[19] An arbitrary constant factor larger than 1 suffices, but we choose 8 to match the formal version of the lemma.

[Zha20]. Let $\{(\Pi_i, I - \Pi_i)\}_i$ be a family of binary-outcome projective measurements indexed by a classical index i of a certain length. Consider the "mixture" M of $\{(\Pi_i, I-\Pi_i)\}_i$, i.e., the procedure that first randomly samples i and then applies the projective measurement $(\Pi_i, I - \Pi_i)$. Zhandry showed the existence of "projective implementation" $\mathsf{ProjImp}$ of M, which is a real-valued projective measurement that "measures" the success probability of M, i.e., for any state ρ, the distribution of applying M on ρ is identical to first applying $\mathsf{ProjImp}$ on ρ to obtain $p \in [0, 1]$ and then outputting 1 with probability p. Though it is unknown how to efficiently implement $\mathsf{ProjImp}$, Zhandry gave an efficient procedure called API ("Approximate Projective Implementation") that approximates $\mathsf{ProjImp}$ in an appropriate sense. Now, we are ready to describe the state repairing technique of [CMSZ21]. Suppose that we apply API on some state, which yields an outcome p, and then apply M. At this point, there is no guarantee on the outcome if we apply API again. The work [CMSZ21] constructed an efficient state repairing procedure Repair which acts on the post-execution state so that the output of API on the resulting state is at least $p-\varepsilon$ with overwhelming probability for an arbitrary noticeable function ε.

Our idea is to apply their technique in our context as follows. We consider a family $\{(\Pi_i, I - \Pi_i)\}_{i\in[n]}$ where Π_i corresponds to the event that $\mathcal{K}_i$ successfully extracts s_i^* and API that approximates the probability that $\mathcal{V}$ returns $\top$. If the initial state is accepted with probability sufficiently larger than γ and ε is set to be sufficiently small, then if we alternately apply $(\Pi_i, I-\Pi_i)$ (i.e., run $\mathcal{K}_i$) and the state repair procedure, we can guarantee that each application of $(\Pi_i, I-\Pi_i)$ results in the first outcome, which corresponds to successfully extracting s_i^*, with probability at least δ. Thus, we can simultaneously extract $s_1^*, \ldots, s_n^*$ if we repeat the above sufficiently many times. Though we eventually prove that this idea works, this is not a direct application of the result of [CMSZ21] since the situation is somewhat different. In particular, we have to make sure that

- we can construct API for any binary-outcome POVMs (that correspond to the success of $\mathcal{V}$), and
- the state repairing procedure still works even if API is defined for a binary-outcome POVM that is irrelevant to the projections $\{(\Pi_i, I - \Pi_i)\}_{i\in[n]}$.

First, we observe that the second point is actually not an issue since this is techncially already proven in [CMSZ21]. That is, even though they only apply their technique in the setting where API is defined for the mixture of projections, their core technical lemma [CMSZ21, Lemma 4.10] already captures the situation where API is irrelevant to those projections. For the first point, though Zhandry showed that an (inefficient) projective implementation can be defined for any binary-outcome POVMs, he did not show how to efficiently approximate it. Thus, we give a construction of API for any binary-outcome POVMs (that correspond to the success of $\mathcal{V}$), which generalizes Zhandry's construction. The construction and its analysis are similar to Zhandry's original one for the case of mixtures of projective measurements while we rely on Jordan's lemma as an additional tool.[20]

2.7 Black-Box Post-Quantum 2PC and MPC with Full Simulation

We begin by recalling the framework established in [CCLY22a], which was devised originally for constant-round black-box PQ-2PC *with ε-simulation*.

A key component of the [CCLY22a] framework is a black-box extractable commit-and-prove protocol with ε-simulation, which we refer to as "ε-ExtCom-n-Prove" henceforth. This primitive enables a committer to commit to a message m during the *Commit Stage* and subsequently prove,

[20] [CMSZ21] also gives a variant of Zhandry's API by using Jordan's lemma, but they also only consider mixtures of projective measurements.

with ε-zero-knowledge, that the committed m satisfies a predicate ϕ during the *Prove Stage*. Furthermore, the Commit Stage itself functions as a post-quantum extractable commitment with ε-simulation, meaning that the post-extraction state of the malicious committer is ε-indistinguishable from that in the real execution. It is worth noting that the symbol ε in the name "ε-ExtCom-n-Prove" indicates that both the zero-knowledge property of the Prove Stage and the post-extraction simulation of the Commit Stage are defined with ε-simulation.

We note that [CCLY22a] can be interpreted as a compiler that transforms a $O(k)$-round black-box ExtCom-n-Prove protocol into a $O(k)$-round black-box PQ-2PC protocol; moreover, if the ExtCom-n-Prove protocol is defined with ε-simulation (resp. full simulation), then the resulting PQ-2PC protocol would be ε-simulatable (resp. fully simulatable). This observation is formalized in Sec. 16.1. Consequently, our goal of constructing $\omega(1)$-round black-box PQ-2PC can be simplified to the task of building $\omega(1)$-round black-box ExtCom-n-Prove protocols with full simulation.

Post-Quantum Extractable Batch Commitments. We further observe that in order to build the desired ExtCom-n-Prove protocols, it (almost) suffices to develop black-box $\omega(1)$-round post-quantum extractable commitments. In this overview, we do not delve into the explanation of why this is true (refer to Sec. 15 for details). More precisely, what we require and refer to as *post-quantum extractable batch commitments* is as follows: The committer is able to commit to a vector of messages $\mathbf{m} = (m_1, \ldots, m_n)$ *collectively*. The commitment can be decommited locally on each index $i \in [n]$. We require the following security to hold:

- **Hiding** For any index $i^* \in [n]$, m_{i^*} remains concealed even if the adversary is given m_i and the corresponding decommitment information for all $i \neq i^*$. In particular, this implies that for any subset $I \subseteq [n]$, the messages corresponding to indices in I remain concealed even if the adversary is given m_i and the corresponding decommitment information for all $i \notin I$.[21]
- **Extractability** There exists a QPT machine $\mathcal{SE}$ (dubbed the *simulation extractor*) capable of extracting the committed vector message $\mathbf{m}^* = (m_1^*, \ldots, m_n^*)$ from the malicious committer C^*, while simultaneously (fully) simulating C^*'s post-extraction state to be negligibly close to that in the real execution between C^* and the honest receiver.

We remark that if we only consider the hiding and binding (rather than extractability), then a simple parallel composition of a stand-alone commitment scheme would suffice. However, extractability may not be preserved under parallel composition, and this is why we need to introduce the above notion of extractable batch commitments.

We manage to build a black-box $\omega(1)$-round construction for such a post-quantum extractable batch commitment, assuming the existence of post-quantum semi-honest oblivious transfer. We provide an overview of this construction in Sec. 2.8. For now, we simply remark that this commitment scheme leads to a black-box $\omega(1)$-round construction of ExtCom-n-prove (with full simulation), which, as discussed earlier, results in the first black-box $\omega(1)$-round PQ-2PC (with full simulation) from the minimal assumption of post-quantum semi-honest oblivious transfers.

Extension to the Multi-Party Setting. Note that the above results for PQ-2PC imply, in particular, a black-box $\omega(1)$-round construction of post-quantum oblivious transfers (maliciously secure, with full simulation). Utilizing a known compiler from [IPS08], such an oblivious transfer protocol can be converted into a black-box PQ-MPC (with full simulation), where the round

[21] This may look similar to *selective-opening security* [Hof11], but we remark that we only require the hiding when I is fixed at the beginning whereas selective-opening security in [Hof11] allows the adversary to adaptively choose I depending on the commitment.

complexity is polynomial in the number of parties (and consequently polynomial in the security parameter λ). We refer to Sec. 16 for details.

2.8 Post-Quantum Extractable Batch Commitments

We now provide an overview of our $\omega(1)$-round construction of post-quantum extractable batch commitments (with full simulation), which only makes black-box use of a constant-round post-quantum semi-honest OT. Since all the primitives in the sequel is post-quantum, we henceforth drop the the quantifier "post-quantum" for succinctness.

We start by describing a protocol that only supports committing to vectors $\mathbf{m}$ of length 1. In this case, the notion of extractable batch commitment degenerates to standard extractable commitments (with full simulation). We emphasize that even such a commitment is previously unknown, if one insists on black-box constructions. Our construction can be divided into the following three steps.

1. Construct $\omega(1)$-round *equivocal* commitments with full simulation based on OWFs. Here, equivocality means that one can simulate the commit stage for malicious receivers in such a way that the commitment can be opened to an arbitrary message in the reveal stage. There are well-known classical black-box constructions of equivocal commitments from OWFs [Kil88, Kil94, PW09], which are later adapted into the post-quantum setting [BCKM21]. Though those constructions are $O(\lambda)$-round, we observe that they can be easily optimized to $\omega(1)$ rounds.

2. Convert equivocal commitments into extractable commitments with a weaker security guarantee which we call *extractability with over-extraction*. It is similar to the standard extractability (with full simulation) except that we allow the extractor to extract a non-$\perp$ message even if the commitment is ill-formed (i.e., there is no valid opening to any message). In fact, we show that our protocol supports (a certain form of) *parallel* extraction with over-extraction, which we elaborate on later. The conversion incurs a constant-round overhead and makes black-box use of constant-round ε-simulatable parallel OT, which in turn is constructed from a constant-round *semi-honest* OT in a black-box manner in [CCLY22a]. We stress that the resulting (parallel) extractable commitment protocol with over-extraction supports full simulation even though the base OT only supports ε-simulation. Since this step is the technical core of our construction of extractable commitments, we will provide more details shortly.

3. Eliminate over-extraction with a $\omega(1)$-round overhead based on OWFs using a standard cut-and-choose technique. Roughly, the commit stage of the protocol works as follows: The committer generates VSS shares of the message m and commits to each share using an extractable commitment scheme with over-extraction in parallel. Then the committer and receiver execute coin-flipping to agree on a subset on which the committer reveals the committed shares along with the corresponding decommitment information. If all of the decommitments are valid, then the receiver is convinced that a large-fraction of the unrevealed commitments is likely to be well-formed (i.e., has a valid decommitment). Then the transcript of the commit stage is well-formed whenever the receiver accepts (except for a negligible probability). In this case, the simulation-extractor can figure out whether the commitment is well-formed by itself and thus over-extraction never occurs. We remark that we only need the coin-flipping protocol to be simulatable against one malicious party (i.e., the role played by the receiver in the commitment protocol), which can be constructed from any equivocal commitments with a constant-round overhead. Since the first step gives $\omega(1)$-round equivocal commitments from OWFs, the overall overhead of round-complexity in this step is just $\omega(1)$.

Below, we give more details of the second step. First, we explain how to achieve extractability with over-extraction in the stand-alone setting (where there is no parallel execution). Our construction works as follows:

Commit stage.

1. The committer C commits to the message m using Naor's commitment.
2. C generates $2k$-out-of-$2k$ secret sharing $\{s_j^b\}_{j\in[k],b\in\{0,1\}}$ of m where $k=\omega(\log\lambda)$. That is, they are uniformly random under the constraint that $\bigoplus_{j\in[k],b\in\{0,1\}} s_j^b = m$.
3. C and the receiver R execute k-parallel execution of ε-simulatable OT where in the j-th execution C uses (s_j^0, s_j^1) as input and R uses an independently random bit r_j as input.
4. C and R engage in the following coin-flipping subprotocol to agree on $t\in\{0,1\}^k$:
   (a) R samples a random string $t_R \leftarrow \{0,1\}^k$ and commits to it using the equivocal commitment scheme.
   (b) C samples a random string $t_C \leftarrow \{0,1\}^k$ and sends it to C.
   (c) R sends to C the value t_R together with the corresponding decommitment information. At this point, C and R agree on $t := t_R \oplus t_C$.
5. C sends $s_j^{t_j}$ for $j\in[k]$ where t_j is the j-th bit of t.

Decommit stage. C sends all the randomness used during the commitment stage as decommitment information, and R accepts if it is consistent to the transcript.

The statistical binding property of the above protocol follows straightforwardly from that of Naor's commitment. The computational hiding property can be shown as follows: Since the OT satisfies ε-simulatable security, for any malicious receiver R^*, we can simulate the execution of the OT in Step 3 only using $\{s_j^{r_j^*}\}_{j\in[k]}$ for some sequence $\{r_j^*\}_{j\in[k]}$ of bits with a noticeable simulation error ε. Since $\{s_j^b\}_{j\in[k],b\in\{0,1\}}$ is $2k$-out-of-$2k$ secret sharing of m, R^* cannot learn any information of m unless t_j happens to be $1-r_j^*$ for all $j\in[k]$. However, by the binding property of the equivocal commitment scheme, R^* can cause only a negligible bias on the distribution of t. Thus, the probability that $t_j = 1-r_j^*$ for all $j\in[k]$ is $2^{-k}+\mathsf{negl}(\lambda)=\mathsf{negl}(\lambda)$. The above argument implies that R^* can distinguish commitments to different messages with advantage at most $\varepsilon+\mathsf{negl}(\lambda)$. Here, ε can be any noticeable function in λ, and thus this actually implies the standard computational hiding.

Below, we give a proof sketch for extractability with over-extraction. We construct the simulation extractor as follows:

- Execute Steps 1 to 3 of the commit stage with the malicious committer C^* while playing the role of the honest receiver. At this point, the simulation extractor obtains $\{s_j^{r_j}\}_{j\in[k]}$ for some random bits r_j.
- Use equivocality to simulate the commit stage of the equivocal commitment scheme in Step 4a.
- Receive t_C from C^* in Step 4b.
- Set $t := (1-r_1)||(1-r_2)||...||(1-r_k)$ and $t_R := t_C \oplus t$. Then open the equvocal commitment in Step 4a to t_R.
- Receive $\{s_j^{t_j} = s_j^{1-r_j}\}_{j\in[k]}$ from C^* in Step 5.
- Output the final state of C^* along with the extracted message $m := \bigoplus_{j\in[k],b\in\{0,1\}} s_j^b$.

It is straightforward to see that the extracted message is equal to the committed message assuming that the transcript is well-formed, i.e., it has a valid opening to some message. Indeed, since the committer is required to reveal all the randomness in the decommit stage, to generate a well-formed transcript, a malicious sender has to follow the protocol albeit with a possibly skewed randomness distribution. In this case, the simulation extractor obtains half of the shares as the output of the OT by the perfect completeness of the OT, and the rest of the shares in the final step. Since the simulation extractor obtains all the secret shares of m, it recovers the correct committed message.

Moreover, we can see that the simulated state of C^* is computationally indistinguishable from the real one (regardless of whether the transcript is well-formed) as follows. We observe that the only difference between the real and simulated execution is that the result t of the coin-flipping is programmed to be $(1-r_1)||(1-r_2)||...||(1-r_k)$ by using equivocality. Though the bits $r_1, ..., r_k$ are also used as the receiver's inputs of the OT, ε-simulatable security of the OT against malicious senders ensure that they are computationally hidden from the view of the malicious committer.[22] Thus, $t_R = t_C \oplus t$ is indistinguishable from uniformly random from the view of the malicious committer. Then we can reduce the indistinguishability between the real and simulated execution to equivocality of the equivocal commitment scheme.

Extractable Batch Commitments. The remaining issue is how to achieve parallel extraction. If we have parallel equivocal commitments, then the above simulation extractor readily extends to the parallel setting. However, the problem is that we do not know how to achieve parallel equivocality in $\omega(1)$ rounds. To circumvent this issue, we change the syntax of the commitment protocol in the parallel setting. That is, instead of considering parallel execution of many copies of the same protocol, we consider a protocol where the committer commits to multiple messages at once, and we require the simulation extractor to extract all the committed messages. In this setting, we can use a single execution of coin-flipping subprotocol to generate the coins (i.e., t in the above protocol) for all the sessions at once. This completely resolves the problem since now there is no parallel execution of the equivocal commitment scheme. In the actual proof, we formalize commitments with such modified syntax as *batch* commitments (see Def. 20) and show that all the remaining steps work with this definition.

3 Preliminaries

3.1 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 $\mathcal{X}$, $x \leftarrow \mathcal{X}$ means that x is uniformly chosen from $\mathcal{X}$.

A function $f : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all polynomial p and sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$; it is said to be *overwhelming* if $1 - f$ is negligible, and said to be *noticeable* if there is a polynomial p such that $f(\lambda) \geq 1/p(\lambda)$ for sufficiently large $\lambda \in \mathbb{N}$. We denote by poly an unspecified polynomial and by negl an unspecified negligible function. For two functions $f_1(\lambda)$ and $f_2(\lambda)$, we will often use $f_1(\lambda) = f_2(\lambda) \pm \mathsf{negl}(\lambda)$ as a shorthand for $|f_1(\lambda) - f_2(\lambda)| \leq \mathsf{negl}(\lambda)$.

Honest (classical) parties are modeled as interactive Turing machines (ITMs). We use PPT and QPT to denote (classical) probabilistic polynomial time and quantum polynomial time, respectively. For a classical probabilistic or quantum algorithm $\mathcal{A}$, $y \leftarrow \mathcal{A}(x)$ means that $\mathcal{A}$ is run on input x and outputs y. When we consider a non-uniform QPT adversary, we specify it by a sequence of polynomial-size quantum circuits with quantum advice $\{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda \in \mathbb{N}}$. In an execution with the

[22] We rely on a well-known fact that ε-simulatable security implies indistinguishability-based security.

security parameter λ, $\mathcal{A}$ runs $\mathcal{A}_\lambda$ taking ρ_λ as the advice. For simplicity, we often omit the index λ and just write $\mathcal{A}(\rho)$ to mean a non-uniform QPT algorithm specified by $\{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$.

Notations for Indistinguishability. We may consider random variables over bit strings or over quantum states. This will be clear from the context. We use the same notations for classical and quantum computational indistinguishability, but there should be no fear of confusion; It means computational indistinguishability against PPT (resp. QPT) distinguishers whenever we consider classical (resp. post-quantum) security. 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 δ, we use $\mathcal{X} \overset{\mathrm{c}}{\approx}_\delta \mathcal{Y}$ to mean that for any non-uniform PPT (resp. QPT) 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

$$|\Pr[\mathcal{A}(X_i)] - \Pr[\mathcal{A}(Y_i)]| \leq \delta(\lambda) + \mathsf{negl}(\lambda). \tag{1}$$

We say that $\mathcal{X}$ and $\mathcal{Y}$ are δ-computationally indistinguishable if the above holds. In particular, when the above holds for $\delta = 0$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable, and simply write $\mathcal{X} \overset{\mathrm{c}}{\approx} \mathcal{Y}$.

Similarly, we use $\mathcal{X} \overset{\mathrm{s}}{\approx}_\delta \mathcal{Y}$ to mean that for any unbounded time algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $i \in I_\lambda$, Inequality (1) holds. In particular, when the above hold for $\delta = 0$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are statistically indistinguishable, and simply write $\mathcal{X} \overset{\mathrm{s}}{\approx} \mathcal{Y}$. Moreover, we write $\mathcal{X} \overset{\mathrm{i.d.}}{\equiv} \mathcal{Y}$ to mean that X_i and Y_i are distributed identically for all $i \in I$.

When we consider an ensemble $\mathcal{X}$ that is only indexed by λ (i.e., $I_\lambda = \{\lambda\}$), we write $\mathcal{X} = \{X_\lambda\}_\lambda$ for simplicity.

3.2 Post-Quantum Commitments

We define (classically-secure and post-quantum) commitments. The following definitions are based on those in [CCLY22a].

Definition 1 (Post-Quantum Commitments). *A* post-quantum commitment scheme $\langle C, R\rangle$ *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 τ,*[23] *C's state ST_C, and R's output $b_{\mathrm{com}} \in \{\bot, \top\}$ indicating acceptance (i.e., $b_{\mathrm{com}} = \top$) or rejection (i.e., $b_{\mathrm{com}} = \bot$). We denote this execution by $(\tau, \mathsf{ST}_C, b_{\mathrm{com}}) \leftarrow \langle C(m), R\rangle(1^\lambda)$. 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 $b_{\mathrm{dec}} \in \{\bot, \top\}$ indicating acceptance (i.e., $b_{\mathrm{dec}} = \top$) or rejection (i.e., $b_{\mathrm{dec}} = \bot$). We assume that R's verification procedure is deterministic and denote it by $\mathsf{Verify}(\tau, m, \mathsf{decom})$.*[24] *W.l.o.g., we assume that R always rejects (i.e., $\mathsf{Verify}(\tau, \cdot, \cdot) = \bot$) whenever $b_{\mathrm{com}} = \bot$. (Note that w.l.o.g.,*

[23] That is, we regard the whole transcript as a commitment.

[24] Note that Verify is well-defined since our syntax does not allow R to keep a state from the commit stage.

τ can include b_{com} because we can always modify the protocol to ask R to send b_{com} as the last round message.)

The scheme satisfies the following requirements:

1. **(Completeness.)** *For any polynomial $\ell : \mathbb{N} \to \mathbb{N}$ and any $m \in \{0,1\}^{\ell(\lambda)}$, it holds that*

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

2. **(Statistically binding.)** *For any unbounded-time committer C^*, the following holds:*

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

3. **(Computationally Hiding.)** *For any non-uniform QPT receiver R^* and any polynomial $\ell : \mathbb{N} \to \mathbb{N}$, the following holds:*

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

where $\mathsf{OUT}_{R^*}\langle C(m_b), R^*\rangle(1^\lambda)$ $(b \in \{0,1\})$ denotes the output of R^* at the end of the commit stage.

For a statistically binding commitment scheme (e.g., the one defined in Def. 1), we often need to talk about the actual value that is "committed" by the committer at the end of the commit stage. For that purpose. we develop a notion in Def. 2 for such a value. Note that this value is not efficiently computable (before the starting of the decommit stage) due to the hiding property of the commitment scheme. Rather, it is simply defined in a information-thoeretical sense.

Definition 2 (Committed Values). *For a statistically binding commitment scheme $\langle C, R\rangle$ (as per Def. 1), we define the value function as follows:*

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

where Verify *is as defined in Def. 1.*

3.3 Post-Quantum Extractable Commitments

We define the post-quantum analog of extractable commitments, which we denote as PQ-ExtCom. As mentioned in the introduction, in the post-quantum setting, we need to explicitly require that the extractor (almost) does not disturb the (potentially malicious) committer's state during the extraction. However, it is not known black-box constructions of such post-quantum extractable commitments exist from (polynomially hard) post-quantum OWFs [CCLY22b]. Fortunately, a recent work [CCLY22a] showed that a constant-round construction from post-quantum OWFs is possible if we relax the extractability to allow an (arbitrarily small) noticeable simulation error. The following definitions are taken from [CCLY22a].

Definition 3 (PQ-ExtCom with ε-Simulation). *A post-quantum commitment scheme $\langle C, R\rangle$ (as per Def. 1) 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 $C^*(\rho)$,*

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

where $\mathsf{val}(\tau)$ is the value committed by C^* as defined in Def. 2.

Parallel Extractability. We also define in Def. 4 the parallel version of Def. 3. This definition considers polynomially many instances of a commitment *in parallel*, where the committers are malicious. It requires the existence of a simulation-extractor $\mathcal{SE}$ that simultaneously extract the values committed in all the sessions, while ε-simulating the post-extraction state of the malicious committers. This definition is a little weak in the sense that it only requires $\mathcal{SE}$ to succeed when R accepts in all the parallel sessions. In particular, when R accepts in some sessions but not in others, the $\mathcal{SE}$ does not need to extract (or simulate) anything. As remarked in [CCLY22a], an alternative stronger (and more natural) definition would require the $\mathcal{SE}$ to extract the committed values in all the sessions where R accepts in the j-th session, and ε-simulate the post-extraction state of the malicious committers across all the sessions (even for those where R rejects at the end of commit stage). However, such a construction *in constant rounds* remains an open challenge, even with non-black-box techniques.

Fortunately, this weak parallel version as per Def. 4 suffices for our purpose.

Definition 4 (Parallel Extractability with ε-Simulation). *A post-quantum commitment scheme $\langle C, R\rangle$ (as per Def. 1) is* parallelly extractable with ε-simulation *if for any integer $n = \mathsf{poly}(\lambda)$, there exists a QPT algorithm $\mathcal{SE}$ (called the parallel ε-simulation extractor) such that for any noticeable $\varepsilon(\lambda)$ and any non-uniform QPT $C^*(\rho)$,*

$$\begin{aligned}&\left\{\mathcal{SE}^{C^*(\rho)}(1^\lambda, 1^{\varepsilon^{-1}})\right\}_\lambda\\ \overset{s}{\approx}_\varepsilon&\left\{\left(\Gamma_{\{b_{\mathrm{com},j}\}_{j=1}^n}(\{\mathsf{val}(\tau_j)\}_{j=1}^n), \mathsf{ST}_{C^*}\right) : (\{\tau_j\}_{j=1}^n, \mathsf{ST}_{C^*}, \{b_{\mathrm{com},j}\}_{j=1}^n) \leftarrow \langle C^*(\rho), R^n\rangle(1^\lambda)\right\}_\lambda\end{aligned}$$

where $(\{\tau_j\}_{j=1}^n, \mathsf{ST}_{C^*}, \{b_{\mathrm{com},j}\}_{j=1}^n) \leftarrow \langle C^*(\rho), R^n\rangle(1^\lambda)$ means that $C^*(\rho)$ interacts with n copies of the honest receiver R in parallel and the execution results in transcripts $\{\tau_j\}_{j=1}^n$, the final state ST_{C^*}, and outputs $\{b_{\mathrm{com},j}\}_{j=1}^n$ of each copy of R and

$$\Gamma_{\{b_{\mathrm{com},j}\}_{j=1}^n}(\{\mathsf{val}(\tau_j)\}_{j=1}^n) := \begin{cases}\{\mathsf{val}_\Pi(\tau_j)\}_{j=1}^n & \text{if } \forall\ j \in [n]\ b_{\mathrm{com},j} = \top\\ \bot & \text{otherwise}\end{cases}.$$

Constant-round and black-box constructions are know for the above versions of post-quantum extractable commitments.

Lemma 3 ([CCLY22a]). *Assume the existence of post-quantum one-way functions, there exist constant-round black-box constructions of:*

- *post-quantum extractable commitments with ε-simulation (as per Def. 3)*
- *post-quatnum parallelly extractable commitments with ε-simulation (as per Def. 4).*

3.4 Post-Quantum Non-Malleable Commitments

We define post-quantum non-malleable commitments (PQ-NMC). Our definition follows the one in [LPY23a], but here we define *1-many* non-malleability directly. We only state the definition in the synchronous setting, supporting polynomially many tags. As mentioned in the introduction, this version suffices for all applications herein.

In fact, we will define and rely on a form of 1-many non-malleability that is weaker than the standard notion, which we accordingly title *weak* (1-many) non-malleability. This bears resemblance to the weak parallel extractability defined above for extractable commitments, in that we will only expect the non-malleability condition to hold in the parallel execution *provided every session of*

this execution is completed successfully (i.e., the receiver in each session accepts the corresponding interaction).

We will start by defining the notion of a man-in-the-middle execution in the 1-many setting, and then present the standard and our weak definition of non-malleability in this setting.

1-Many Man-in-the-Middle Execution. Let $\langle C, R\rangle$ be a statistically binding and computationally hiding post-quantum commitment scheme. We use a *tag-based* specification so that every execution of $\langle C, R\rangle$ is associated with a tag $t \in [T]$, where T is an integer. Consider a non-uniform QPT adversary $\mathcal{M} = \{\mathcal{M}_\lambda, \rho_\lambda\}_\lambda$ participating in $(k+1)$ instances of $\langle C, R\rangle$ as follows: $\mathcal{M}_\lambda(\rho_\lambda)$ plays the role of the receiver in one instance (referred to as the *left session*), while simultaneously acting as a committer in the other k sessions (referred to as the *right sessions*). *All the* $(k+1)$ *sessions are execute in parallel*, and we refer to this setting as the *synchronous 1-k MIM execution*, where "MIM" is the acronym for "man-in-the-middle."

Notation-wise, we denote the relevant entities used in the right interaction as the "tilde'd" version of the corresponding entities on the left. In particular, let t denote the tag associated with the left session and $(\widetilde{t}_1, \ldots, \widetilde{t}_k)$ denote the tags for the k right sessions respectively; let m denote the value committed by the honest C in the left session, and $(\widetilde{m}_1, \ldots, \widetilde{m}_k)$ the values committed by $\mathcal{M}_\lambda(\rho_\lambda)$ in the k right sessions respectively, i.e., we set $\widetilde{m}_i = \mathsf{val}(\widetilde{\tau}_i)$ where $\widetilde{\tau}_i$ is the transcript of the i-th right session (see Def. 2).

For this 1-k MIM execution, let $\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m, \rho_\lambda)$ denote concatenation of the final output of $\mathcal{M}_\lambda(\rho_\lambda)$ and the values committed in all the k right sessions, *when the honest* C *in the left session commits to value* m. That is,

$$\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m, \rho_\lambda) \coloneqq \big(\mathsf{OUT}_{\mathcal{M}}, (\widetilde{m}_1, \ldots, \widetilde{m}_k)\big).$$

Definition 5 (Standard Synchronous 1-Many PQ-NMC). *A post-quantum statistically binding commitment* $\langle C, R\rangle$ *is said to be 1-k non-malleable if for all polynomial* $\ell(\cdot)$ *and all non-uniform QPT adversaries* $\mathcal{M} = \{\mathcal{M}_\lambda, \rho_\lambda\}_\lambda$ *participating the above synchronous 1-k MIM execution with* $t \neq \widetilde{t}_i$ *for all* $i \in [k]$*, it holds that*

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

Some remarks follow:

1. Def. 5 requires that the left-session tag t is different from those for all right sessions. This is standard practice when defining non-malleability, with the purpose of ruling out the uninteresting case when $\mathcal{M}$ is simply acting as a channel, forwarding messages from C on the left to the R on some right session.
2. Def. 5 does not consider entanglement between $\mathcal{M}$'s auxiliary input and distinguisher's auxiliary input. However, [BLS22, Claim 3.1] shows that the above definition implies the version that considers such entanglement.
3. **(1-1 Non-Malleability.)** When $k = 1$, Def. 5 degenerates to the standard definition of post-quantum non-malleability (in the synchronous setting).

Next we turn to our weaker definition of 1-many non-malleability.

Definition 6 (Weak 1-Many PQ-NMC). *Let* $\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m, \rho_\lambda)$ *denote the output of the synchronous 1-k man-in-the-middle execution as above, and let* $d_j \in \{\top, \bot\}$ *denote the decision of*

the receiver in each session for $j \in [k]$. Define the function $\Gamma_{\{d_j\}_{j=1}^k}(\cdot)$ as follows:

$$\Gamma_{\{d_j\}_{j=1}^k}(\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m,\rho_\lambda)) := \begin{cases} \mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m,\rho_\lambda) & \text{if } \forall\ j \in [k],\ d_j = \top \\ (\mathsf{OUT}_{\mathcal{M}}, \bot^k) & \text{otherwise} \end{cases},$$

where $\mathsf{OUT}_{\mathcal{M}}$ is the first component of $\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m,\rho_\lambda)$.

A post-quantum statistically binding commitment $\langle C, R\rangle$ is said to be weakly *1-k non-malleable if for all polynomial $\ell(\cdot)$ and all non-uniform QPT adversaries $\mathcal{M} = \{\mathcal{M}_\lambda, \rho_\lambda\}_\lambda$ participating the above synchronous 1-k MIM execution with $t \neq \widetilde{t}_i$ for all $i \in [k]$, it holds that*

$$\begin{aligned} &\left\{\Gamma_{\{d_j\}_{j=0}^k}(\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m_0,\rho_\lambda))\right\}_{\lambda\in\mathbb{N},m_0,m_1\in\{0,1\}^{\ell(\lambda)}} \\ \overset{c}{\approx}\ &\left\{\Gamma_{\{d_j\}_{j=0}^k}(\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m_1,\rho_\lambda))\right\}_{\lambda\in\mathbb{N},m_0,m_1\in\{0,1\}^{\ell(\lambda)}}. \end{aligned}$$

Remark 1. Note that Def. 5 and Def. 6 are in fact *equivalent* in the basic 1-1 setting (i.e., the man-in-the-middle runs exactly one left and right session each). This is easily verified by observing the definition of the output in the MIM experiment: it is clear to see that in cases where the right interaction is completed successfully, the definitions are identical. On the other hand, when the right interaction is not completed successfully, the output according to both Def. 5 and Def. 6 consists of the output state of the MIM and the $\bot$ symbol.

Remark 2. We further observe that in analogy with the standard definition, weak one-many non-malleability also implies weak *many-many* non-malleability (which can be defined analogously). As in the standard case, this can be easily inferred from a standard hybrid argument (where the input in each left session is switched in turn).

3.5 Post-Quantum MPC with ε-Simulation

We present the formal definition for PQ-MPC with ε-simulation. It is identical to the standard MPC definition in the classical setting except that:

1. The malicious party can be a QPT machine;
2. The indistinguishability between the real-world execution and the simulated one is parameterized by a noticeable function $\varepsilon(\lambda)$.

Consider n parties $P_1, \ldots, P_n$ who wish to interact in a protocol Π to evaluate a n-party classical functionality f on their joint inputs. They 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.

In this work, we consider a *static* adversary, namely, at the beginning of the execution the adversary specifies a set I of corrupted parties which she controls, and through the execution she will not change the set I. The ideal and real executions follow the standard description as in, e.g., [Gol04, Lin16].

In the real world, a non-uniformal QPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda, \rho_\lambda\}_\lambda$ corrupting $\{P_i\}_{i\in I}$ interacts with $\{P_i\}_{i\in[n]\setminus I}$. Let $\mathbf{x} = (x_1, \ldots, x_n)$ denote the respective initial input to each party. Let $\mathsf{REAL}_{\Pi,\mathcal{A}_\lambda,I}(\lambda, \mathbf{x}, \rho_\lambda)$ denote the random variable consisting of the output of the adversary (which

may be an arbitrary function of its view and in particular may be a quantum state) *and* the outputs of the uncorrupted parties $\{P_i\}_{i\in[n]\setminus I}$.

In the ideal world, a QPT machine $\mathcal{S}$ controls the same parties in I as $\mathcal{A}_\lambda$. It gets $\{x_i\}_{i\in I}$ as input and is granted black-box access to $\mathcal{A}_\lambda(\rho_\lambda)$. Similar as in the ε-ZK definition [CCY21, CCLY22a], $\mathcal{S}$ additionally takes as input a "slackness parameter" $\varepsilon(\lambda)$, which is a noticeable function on λ. Henceforth, we always require that $\mathcal{S}$'s running time is a polynomial on both λ and ε^{-1}. In this ideal-world execution, let $\mathsf{IDEAL}_{f,\mathcal{S},I}(\lambda,\varepsilon,\mathbf{x},\rho_\lambda)$ denote the outputs of $\mathcal{S}$ (with slackness ε) and the outputs of the uncorrupted parties $\{P_i\}_{i\in[n]\setminus I}$.

Definition 7 (Post-Quantum MPC with ε-Simulation). *Let f be a classical n-party functionality, and Π be a classical n-party protocol. We say that Π is a post-quantum MPC protocol for f with ε-simulation if there exists a QPT simulator $\mathcal{S}$ such that for any non-uniform QPT adversary $\mathcal{A}=\{\mathcal{A}_\lambda,\rho_\lambda\}_{\lambda\in\mathbb{N}}$, any $I\subset[n]$, any $\mathbf{x}\in(\{0,1\}^*)^n$, and any noticeable function $\varepsilon(\lambda)$, it holds that:*

$$\{\mathsf{REAL}_{\Pi,\mathcal{A}_\lambda,I}(\lambda,\mathbf{x},\rho_\lambda)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx}_\varepsilon \{\mathsf{IDEAL}_{f,\mathcal{S},I}(\lambda,\varepsilon,\mathbf{x},\rho_\lambda)\}_{\lambda\in\mathbb{N}}.$$

3.6 Verifiable Secret Sharing and Information-Theoretic MPC

Verifiable Secret Sharing. We present in Def. 8 the definition of verifiable secret sharing (VSS) schemes [CGMA85]. We remark that [BGW88, CDD+99] implemented $(n+1,\lfloor n/3\rfloor)$-perfectly secure VSS schemes. These constructions suffice for all the applications in the current paper.

Definition 8 (Verifiable Secret Sharing). *An $(n+1,t)$-perfectly secure VSS scheme Π_{VSS} consists of a pair of protocols $(\mathsf{VSS}_{\mathsf{Share}},\mathsf{VSS}_{\mathsf{Recon}})$ that implement respectively the sharing and reconstruction phases as follows.*

- **Sharing Phase $\mathsf{VSS}_{\mathsf{Share}}$:** *Player P_{n+1} (referred to as dealer) runs on input a secret s and randomness r_{n+1}, while any other player P_i ($i\in[n]$) runs on input a randomness r_i. During this phase players can send (both private and broadcast) messages in multiple rounds.*
- **Reconstruction Phase $\mathsf{VSS}_{\mathsf{Recon}}$:** *Each shareholder sends its view v_i ($i\in[n]$) of the Sharing Phase to each other player, and on input the views of all players (that can include bad or empty views) each player outputs a reconstruction of the secret s.*

All computations performed by honest players are efficient. The computationally unbounded adversary can corrupt up to t players that can deviate from the above procedures. The following security properties hold.

1. **Perfectly Verifiable-Committing:** *if the dealer is dishonest, then one of the following two cases happen (i.e., with probability 1):*
   (a) *During the Sharing Phase, honest players disqualify the dealer, therefore they output a special value $\perp$ and will refuse to play the reconstruction phase;*
   (b) *During the Sharing Phase, honest players do not disqualify the dealer. Therefore such a phase determines a unique value s^* that belongs to the set of possible legal values that does not include $\perp$, which will be reconstructed by the honest players during the reconstruction phase.*
2. **Secrecy:** *if the dealer is honest, then the adversary obtains no information about the shared secret before running the protocol* Recon. *More accurately, there exists a PPT oracle machine $\mathcal{S}^{(\cdot)}$ such that for any message m, and every (potentially inefficient) adversary $\mathcal{A}$ corrupting a*

set T of parties with $|T| \leq t$ during the Sharing Phase $\mathsf{VSS}_{\mathsf{Share}}(m)$ (denote $\mathcal{A}$'s view in this execution as $\mathsf{View}_{\mathcal{A},T}(1^\lambda, m)$), the following holds: $\{\mathsf{View}_{\mathcal{A},T}(1^\lambda, m)\} \stackrel{i.d.}{\equiv} \{\mathcal{S}^{\mathcal{A}}(1^\lambda, T)\}$.

3. **Correctness:** *if the dealer is honest throughout the protocols, then each honest player will output the shared secret s at the end of protocol* Recon.

Information-Theoretically Secure MPC. We first recall *information-theoretically secure* MPC and relevant notions that will be employed in the MPC-in-the-head paradigm shown later.

Information-Theoretic MPC. We now define MPC in the information-theoretic setting (i.e., secure against unbounded adversaries).

Definition 9 (Perfectly/Statistically-Secure MPC). *Let $f : (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ be an n-ary functionality, and let Π be a protocol. We say that Π* (n,t)-perfectly (resp., statistically) securely *computes f if for every static, malicious, and (possibly-inefficient) probabilistic adversary $\mathcal{A}$ in the real model, there exists a probabilistic adversary $\mathcal{S}$ of comparable complexity (i.e., with running time polynomial in that of $\mathcal{A}$) in the ideal model, such that for every $I \subset [n]$ of cardinality at most t, every $\mathbf{x} = (x_1, \ldots, x_n) \in (\{0,1\}^*)^n$ (where $|x_1| = \cdots = |x_n|$), and every $z \in \{0,1\}^*$, it holds that:*

$$\{\mathsf{REAL}_{\Pi,\mathcal{A}(z),I}(\mathbf{x})\} \stackrel{i.d.}{\equiv} \{\mathsf{IDEAL}_{f,\mathcal{S}(z),I}(\mathbf{x})\} \quad \big(\textit{resp.}, \{\mathsf{REAL}_{\Pi,\mathcal{A}(z),I}(\mathbf{x})\} \stackrel{s}{\approx} \{\mathsf{IDEAL}_{f,\mathcal{S}(z),I}(\mathbf{x})\}\big).$$

Recall that the MPC protocol from [BGW88] achieves (n,t)-perfect security (against static and malicious adversaries) with t being a constant fraction of n.

Theorem 5 ([BGW88]). *Consider a synchronous network with pairwise private channels. Then, for every n-ary functionality f, there exists a protocol that (n,t)-perfectly securely computes f in the presence of a static malicious adversary for any $t < n/3$.*

Consistency, Privacy, and Robustness. We now define some notation related to MPC protocols. Their roles will become clear when we discuss the MPC-in-the-head technique later.

Definition 10 (View Consistency). *A view View_i of an honest player P_i during an MPC computation Π contains input and randomness used in the computation, and all messages received from and sent to the communication tapes. A pair of views $(\mathsf{View}_i, \mathsf{View}_j)$ is* consistent *with each other if*

1. *Both corresponding players P_i and P_j individually computed each outgoing message honestly by using the random tapes, inputs and incoming messages specified in View_i and View_j respectively, and:*
2. *All output messages of P_i to P_j appearing in View_i are consistent with incoming messages of P_j received from P_i appearing in View_j (and vice versa).*

Remark 3 (View Consistency of VSS). Although Def. 10 defines view consistency for MPC protocols, we will also refer to the view consistency for the execution of verifiable secret sharing schemes (Def. 8). The views $(\mathsf{v}_i, \mathsf{v}_j)$ of players i and j (excluding the dealer) during the execution of $\mathsf{VSS}_{\mathsf{Share}}$ is said to be consistent if any only if $(\mathsf{v}_i, \mathsf{v}_j)$ satisfies the two requirements in Def. 10.

We further define the notions of correctness, privacy, and robustness for multi-party protocols.

Definition 11 (Semi-Honest Computational Privacy). *Let $1 \leq t < n$, let Π be an MPC protocol, and let $\mathcal{A}$ be any* static, PPT, and semi-honest *adversary. We say that Π realizes a function $f : (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ with* semi-honest (n,t)-computational privacy *if there is a PPT simulator $\mathcal{S}$ such that for any inputs $x, w_1, \ldots, w_n$, every subset $T \subset [n]$ ($|T| \leq t$) of players corrupted by $\mathcal{A}$, and every D with circuit size at most $\mathsf{poly}(\lambda)$, it holds that*

$$\left| \Pr[D(\mathsf{View}_T(x, w_1, \ldots, w_n)) = 1] - \Pr[D(\mathcal{S}(T, x, \{w_i\}_{i \in T}, f_T(x, w_1, \ldots, w_n))) = 1] \right| \leq \mathsf{negl}(\lambda), \quad (2)$$

where $\mathsf{View}_T(x, w_1, \ldots, w_n)$ is the joint view of all players.

Definition 12 (Statistical/Perfect Correctness). *Let Π be an MPC protocol. We say that Π realizes a deterministic n-party functionality $f(x, w_1, \ldots, w_n)$ with* perfect (resp., statistical) *correctness if for all inputs $x, w_1, \ldots, w_n$, the probability that the output of some party is different from the output of f is 0 (resp., negligible in k), where the probability is over the independent choices of the random inputs $r_1, \ldots, r_n$ of these parties.*

Definition 13 (Perfect/Statistical Robustness). *Assume the same setting as the previous definition. We say that Π realizes f with (n,t)-*perfect (resp., statistical) robustness *if in addition to being perfectly (resp., statistical) correct in the presence of a semi-honest adversary as above, it enjoys the following* robustness *property against any computationally unbounded malicious adversary corrupting a set T of at most t parties, and for any inputs $(x, w_1, \ldots, w_n)$: if there is no $(w'_1, \ldots, w'_n)$ such that $f(x, w'_1, \ldots, w'_n) = 1$, then the probability that some uncorrupted player outputs 1 in an execution of Π in which the inputs of the honest parties are consistent with $(x, w_1, \ldots, w_n)$ is 0 (resp., negligible in λ).*

3.7 MPC-in-the-Head

MPC-in-the-head (MitH) is a technique originally developed for constructing black-box ZK protocols from MPC protocols [IKOS07]. Intuitively, the MPC-in-the-head idea works as follows. Let $\mathcal{F}_{\text{ZK}}$ be the zero-knowledge functionality for an **NP** language. Assume there are n parties holding a witness in a secret-sharing form. $\mathcal{F}_{\text{ZK}}$ takes as public input x and one share from each party, and outputs 1 iff the secret reconstructed from the shares is a valid witness. To build a ZK protocol, the prover runs in his head an execution of MPC w.r.t. $\mathcal{F}_{\text{ZK}}$ among n imaginary parties, each one participating in the protocol with a share of the witness. Then, it commits to the view of each party separately. The verifier obtains t randomly chosen views, checks that such views are "consistent" (see Def. 10), and accepts if the output of every party is 1. The idea is that, by selecting the t views at random, V will catch inconsistent views if the prover cheats.

We emphasize that, in this paradigm, a malicious prover decides the randomness of each virtual party, including those not checked by the verifier (corresponding to honest parties in the MPC execution). Therefore, MPC protocols with standard computational security may fail to protect against such attacks. We need to ensure that the adversary cannot force a wrong output even if it additionally controls the honest parties' random tapes. The $(n, \lfloor n/3 \rfloor)$-perfectly secure MPC protocol in Thm. 5 suffices for this purpose (see also Rmk. 4).

One can extend this technique further (as in [GLOV12]), to prove a general predicate ϕ about an arbitrary value α. Namely, one can consider the functionality $\mathcal{F}_\phi$ in which party i participates with input a VSS share $[\alpha]_i$. $\mathcal{F}_\phi$ collects all such shares, and outputs 1 iff $\phi(\mathsf{VSS}_{\mathsf{Recon}}([\alpha]_1, \ldots, [\alpha]_n)) = 1$.

Remark 4 (Exact Security Requirements on the Underlying MPC.). To be more accurate, any MPC protocol that achieves *semi-honest (n,t)-computational privacy (as per Def. 11) and (n,t)-perfect robustness (as per Def. 13)* will suffice for the MPC-in-the-head application.[25] These two

[25] It is also worth noting that the (n,t)-perfect robustness could be replaced with *adaptive (n,t)-statistical robustness*. See [IKOS07, Section 4.2] for more details.

requirements are satisfied by any (n,t)-perfectly secure MPC (and, in particular, the one from Thm. 5).

MPC-in-the-Head Commitments. We present a hiding game that relies on the (MitH) technique, and show that no QPT adversary can win this game with non-negligible probability. This game is essentially a black-box post-quantum commitment protocol due to [GLOV12, CCLY22a]. We choose to present it as the following hiding game because this game will be of direct use later when proving the security of our protocol in Sec. 5.3 (particularlly in Lem. 10).

Experiment 1: VSS Hiding Game

Parameters: Let $n(\lambda)$ be a polynomial in λ. Let k be a constant-fraction of n such that $k \leq \frac{n}{3}$. This involves an (efficient) challenger Ch interacting with the adversary $\mathcal{A}$. The interaction proceeds as follows:

1. $\mathcal{A}$ selects messages $m_0, m_1 \in \{0,1\}^\lambda$ and sends these to Ch.
2. Next, $\mathcal{A}$ samples a random size-k subset η of $[n]$. It then runs an interaction of ExtCom (as per Def. 3) with Ch, where it commits to η.
3. Ch prepares n views $\{\mathsf{v}_i\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the message m_b (see Rmk. 5 for details). Ch commits to each v_i $(i \in [n])$ independently in parallel, using Naor's commitment.

   Remark 5. We describe this step more explicitly. Ch emulates $n+1$ virtual parties $\{P_i\}_{i\in[n+1]}$ 'in its head.' Party P_{n+1} is the dealer, possessing the string x. Other parties do not have any input. These parties execute the $\mathsf{VSS}_{\mathsf{Share}}$ stage of the $(n+1,k)$-VSS scheme to compute the functionality $\mathsf{VSS}_{\mathsf{Share}}$. At the end of the execution, P_i $(i \in [n])$ obtains the i-th VSS share of m_b as the output, and P_{n+1} does not receive any output. The $\{\mathsf{v}_i\}_{i\in[n]}$ corresponds to the views of $\{P_i\}_{i\in[n]}$ from this execution (emulated in Ch's head).
4. $\mathcal{A}$ sends η togther with the decommitment informaiotn w.r.t. the ExtCom in Step 2.
5. Ch then decommits to the VSS shares in the set η, i.e. it sends $\{\mathsf{v}_i\}_{i\in\eta}$ along with the corresponding decommitment information w.r.t. the commitment in Step 3.
6. Finally, $\mathcal{A}$ submits a guess bit b' corresponding to its estimate of which message was committed to by Ch.

Output: We use $\mathsf{VSS}_{\mathsf{hd}}(1^\lambda, \mathcal{A})$ to denote the output of this game, where $\mathsf{VSS}_{\mathsf{hd}}(\mathcal{A}) = 1$ iff $b' = b$.

Lemma 4. *For any QPT adversary $\mathcal{A}$, it holds that* $\Pr\left[\mathsf{VSS}_{\mathsf{hd}}(1^\lambda, \mathcal{A}) = 1\right] = \frac{1}{2} \pm \mathsf{negl}(\lambda)$, *where* $\mathsf{VSS}_{\mathsf{hd}}(1^\lambda, \mathcal{A})$ *is defined in Expr. 1.*

Proof Sketch. The proof of this lemma already appears in [CCLY22a, Section 6.5]. Here, we only recall the high-level idea. We will show the when Ch changes the committed value from m_0 to m_1, $\mathcal{A}$ cannot tell the difference. For this, we assume for contradiction that $\mathcal{A}$ can tell the different with some inverse-polynomial advantage $\delta(\lambda)$ for infinitely many $\lambda \in \mathbb{N}$. Then, Ch can extract the subset η from Step 2, using the simulation-extractor $\mathcal{SE}$ guaranteed by Def. 3, setting the error parameter $\varepsilon := \frac{\delta}{3}$.

With the η in hand, Ch does not need to generate the views $\{\mathsf{v}_i\}_{i\in[n]}$ in Step 3 honestly. Instead, it can invoke the (n,k)-MitH simulator to simulate the views in set η, and set other views $\{\mathsf{v}_i\}_{i\in[n]\setminus\eta}$

to all-0 strings with proper length. By the security of the underlying (n, k)-MPC, the views in η does not contain any information of the committed value m_b. This helps Ch to change the committed value from m_0 to m_1.

By our parameter setting $\varepsilon := \frac{\delta}{3}$, after Ch changes the committed value from m_0 to m_1, $\mathcal{A}$ can tell the different with probability at most $\frac{2\delta}{3}$, which is still smaller than δ, reaching the desired contradiction.

□

MPC-in-the-Head Interactive Arguments. As a proof of concept, we show in the following a constant-round black-box interactive argument built from the MitH technique. This protocol is taken from [IKOS07, CCLY22a]. A prover C first commits to a string x, and then starts to interact with the verifier R for the statement that the committed x satisfies a predicate $\phi(\cdot)$. The soundness requirement is: if $\phi(x) = 0$, then the verifier will reject the proof except for negligible probability.

Protocol 1: MPC-in-the-Head Interactive Argument

Parameters: Let n be a polynomial in λ, and k be a constant fraction of n such that $k \leq n/3$. We will employ a $(n+1, k)$ VSS scheme and a (n, k)-secure MPC scheme.
Inputs: Both parties receive λ as the common input. The prover in addition gets a string x as its private input.

Commit Stage: In this stage, C commits to the string x (using the MitH approach).

- C prepares n views $\{\mathsf{v}_i\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1, k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string x (see Rmk. 5 for details). C commits to each v_i $(i \in [n])$ independently in parallel, using Naor's commitment.

Proof Stage: Both parties learn an efficiently computable predicate $\phi(\cdot)$.

1. C then prepares n views $\{\mathsf{v}'_i\}_{i\in[n]}$ corresponding to an (n, k)-MitH execution for the functionality F_ϕ described below, where party P_i uses v_i as input. It then commits to each of these views v'_i independently in parallel using Naor's commitment.
   - **Functionality F_ϕ:** This collects inputs v_i from party i, runs $\mathsf{VSS}_{\mathsf{Recon}}$ on these inputs to recover a value x, and outputs $\phi(x)$.
2. R then samples a size-k random subset $\eta \subset [n]$ and sends it to C.
3. C now decommits to the views $\{\mathsf{v}_i\}_{i\in\eta}$ and $\{\mathsf{v}'_i\}_{i\in\eta}$.
4. Finally, R checks these decommitments, and also checks if these revealed views are *consistent* w.r.t. the VSS and MPC executions, here by 'consistent' we refer to the consistency requirements as per Def. 10 and Rmk. 3. It also checks for each $i \in \eta$ the final output of P_i contained in v'_i is 1. It aborts immediately if any of the checks fail.

We know state the properties Prot. 1 satisfies as Lem. 5.

Lemma 5 ([IKOS07, CCLY22a].). *Prot. 1 satisfies the following properties:*

1. *The* **Commit Stage** *is a statistically binding commitment.*
2. *If the x committed in the* **Commit Stage** *satisfies $\phi(x) = 1$, then R accepts with probability 1.*
3. *If the x committed in the* **Commit Stage** *satisfies $\phi(x) = 0$, then R rejects except for with probability $O(2^k)$, even if C^* is a malicious QPT machine and ϕ is picked by C^*.*

Extension of Prot. 1. We remark that [IKOS07, CCLY22a] indeed proved that Prot. 1 can be converted into a ε-simulatable zero-knowledge protocol if Step 2 is replaced with a coin-flipping protocol that is ε-simultable against any QPT R^*. Such a coin-flipping protocol can be constructed as follows: (1) R^* to commit to a share η_R using a extractable commitment with ε-simulation (as per Def. 3); (2) C sends a random share η_C; (3) R decommits to η_R. The coin-flipping result will be $\eta := \eta_C \oplus \eta_R$.

3.8 Watrous' Rewinding Lemma

The following is Watrous' rewinding lemma [Wat09] in the form of [BS20, Lemma 2.1].

Lemma 6 (Watrous' Rewinding Lemma [Wat09]). *There is a quantum algorithm* R *that gets as input the following:*

- *A quantum circuit* Q *that takes* n*-input qubits in register* Inp *and outputs a classical bit* b *and an* m*-qubit output.*
- *An* n*-qubit state* ρ *in register* Inp*.*
- *A number* $T \in \mathbb{N}$ *in unary.*

$\mathsf{R}(1^T, \mathsf{Q}, \rho)$ *executes in time* $T \cdot |\mathsf{Q}|$ *and outputs a distribution over* m*-qubit states* $D_\rho := \mathsf{R}(1^T, \mathsf{Q}, \rho)$ *with the following guarantees.*

For an n*-qubit state* ρ*, denote by* Q^0_ρ *the conditional distribution of the output distribution* $\mathsf{Q}(\rho)$*, conditioned on* $b = 0$*, and denote by* $p(\rho)$ *the probability that* $b = 0$*. If there exist* $p_0, q \in (0,1)$*,* $\gamma \in (0, \frac{1}{2})$ *such that:*

- *Amplification executes for enough time:* $T \geq \frac{\log(1/\gamma)}{4p_0(1-p_0)}$*,*
- *There is some minimal probability that* $b = 0$*: For every* n*-qubit state* ρ*,* $p_0 \leq p(\rho)$*,*
- $p(\rho)$ *is input-independent, up to* γ *distance: For every* n*-qubit state* ρ*,* $|p(\rho) - q| < \gamma$*, and*
- q *is closer to* $\frac{1}{2}$*:* $p_0(1-p_0) \leq q(1-q)$*,*

then for every n*-qubit state* ρ*,*

$$\mathsf{TD}(\mathsf{Q}^0_\rho, D_\rho) \leq 4\sqrt{\gamma}\frac{\log(1/\gamma)}{p_0(1-p_0)}.$$

Moreover, $\mathsf{R}(1^T, \mathsf{Q}, \rho)$ *only makes black-box use of* $\mathsf{Q}(\rho)$*.*[26]

4 Post-Quantum Trapdoor Coin-Tossing with ε-Simulation

In this section, we build a *trapdoor coin-flipping* protocol. This is a coin-flipping protocol between two parties C and R where C additionally commits to some string x before the coin-flipping starts. The security guarantee of the actual coin-flipping stage is 'controlled' by the committed string x and a predicate $\phi(\cdot)$ that is given to both party at the beginning of the coin-flipping stage. In particular, if the committed x does not satisfy the predicate $\phi(\cdot)$, then even a cheating C^* cannot gain any advantage over R in the coin-flipping stage. This is formalized as the standard simulation-based requirement, namely, we require the existence of a simulator that can 'enforce' the coin-tossing result to a given random string if C^* does not abort the execution.

[26] The black-boxness is observed in [CCY21, CCLY22a]

On the other hand, if the committed x satisfies the predicate, then C can enforce the coin-tossing result to a given random string against any efficient R^* that does not abort the execution (this is what we call 'trapdoor'). We require that this 'enforcing procedure' should happen in straight-line, i.e., it is not allowed to rewind R^*. Indeed, we actually require a stronger version of this property as follows.

We consider a real-world execution where C commits honestly to a string x to the malicious R^*; Here, we do not require that x satisfies ϕ. Then, in the simulation world, we provide a potentially different x' to a 'straight-line' simulator $\mathcal{S}$ such that $\phi(x') = 1$. With this valid witness x', $\mathcal{S}$ can enforce the coin-flipping result to a pre-sampled random string, interacting in straight-line with a potentially malicious R^*.

In the following, we first present the definition in Sec. 4.1 and then show our construction with security proof in Sec. 4.2. We remark that this construction is a simple application of the MPC-in-the-Head technique. Indeed, the main contribution of this work does not lie in this construction. We abstract out the notion of 'trapdoor coin-flipping' simply because it helps us present our post-quantum non-malleable commitments (in Sec. 5) in a modular manner.

4.1 Definition

Definition 14. *A post-quantum trapdoor coin-tossing with ε-simulation protocol consists of a pair of* PPT *ITM $\langle C, R\rangle$. Let $x \in \{0,1\}^{\ell(\lambda)}$ (where $\ell(\cdot)$ is some polynomial) is a message that C wants to commit to. The protocol consists of the following stages (we omit the input 1^λ to C and R):*

- **Commit Stage:** *$C(x)$ and R interacts to generates a transcript (commitment) com, C's state ST_C, and R's decision bit $b \in \{\top, \bot\}$ indicating acceptance (i.e., $b = \top$) or rejection (i.e., $b = \bot$). We denote this execution as $(\mathsf{com}, \mathsf{ST}_C, b) \leftarrow \langle C(x), R\rangle_{COM}$. Note that a malicious receiver is allowed to output any quantum state, which we denote by ST_{R^*} instead of b, and to keep the state for the following stages.*
- **Decommit Stage:**[27] *$C(\mathsf{ST}_C)$ generates a decommitment decom and sends it to R along with a message x. R accepts or rejects.*
- **Coin-Flipping Stage:** *Let $\phi(\cdot)$ be any predicate. $C(\mathsf{ST}_C, \phi)$ and $R(\mathsf{com}, \phi)$ interacts, after which R outputs $\top$ (accept) or $\bot$ (reject). We denote the execution of this stage as $(\eta_1, \eta_2) \leftarrow \langle C(\mathsf{ST}_C), R(\mathsf{com})\rangle^{\phi}_{CF}$, where $\eta_1, \eta_2 \in \{0,1\}^*$ are the respective output of C and R. Note that a malicious receiver is allowed to output any quantum state, which we denote by OUT_{R^*} instead of η_2.*

The following requirements are satisfied:

1. **Statistically Binding.** *The Commit Stage and Decommit Stage together constitute a post-quantum commitment scheme that is statistically binding and (post-quantum) computational hiding.*
2. **Completeness.** *For any $x \in \{0,1\}^{\ell(\lambda)}$ and any predicate ϕ, it holds that*

$$\Pr\left[\eta_1 = \eta_2 \ : \ \begin{array}{l}(\mathsf{com}, \mathsf{ST}_C, b) \leftarrow \langle C(x), R\rangle_{COM} \\ (\eta_1, \eta_2) \leftarrow \langle C(\mathsf{ST}_C), R(\mathsf{com})\rangle^{\phi}_{CF}\end{array}\right] = 1. \tag{3}$$

3. **Security against Malicious R^*.** *For any QPT R^*, there exists a 'straight-line' QPT simulator $\mathcal{S}$ such that for any efficiently computable predicate $\phi(\cdot)$, any $x, x' \in \{0,1\}^{\ell(\lambda)}$ such that $\phi(x') = 1$,*

[27] This stage is rarely executed in applications.

it holds that

$$\left\{(\eta_1,\mathsf{OUT}_{R^*}) \;:\; \begin{array}{l}(\mathsf{com},\mathsf{ST}_C,\mathsf{ST}_{R^*}) \leftarrow \langle C(x),R^*\rangle_{COM}\\ (\eta_1,\mathsf{OUT}_{R^*}) \leftarrow \langle C(\mathsf{ST}_C),R^*(\mathsf{ST}_{R^*})\rangle^{\phi}_{CF}\end{array}\right\}_\lambda \overset{c}{\approx} \left\{(\Gamma_d(\eta),\mathsf{OUT}_{R^*}) \;:\; \begin{array}{l}(\mathsf{ST}_S,\mathsf{ST}_{R^*}) \leftarrow \langle \mathcal{S}(x'),R^*\rangle_{COM}\\ \eta \overset{\$}{\leftarrow} \{0,1\}^{k(\lambda)}\\ (d,\mathsf{OUT}_{R^*}) \leftarrow \langle \mathcal{S}(\mathsf{ST}_S,\eta),R^*(\mathsf{ST}_{R^*})\rangle^{\phi}_{CF}\end{array}\right\}_\lambda,$$

where $\Gamma_d(\eta) \coloneqq \begin{cases}\eta & \text{if } d=\top\\ \bot & \text{if } d=\bot\end{cases}$.

4. **Security against Malicious C^*.** *For any QPT C^* with ST_{C^*} there exists a QPT simulator $\mathcal{S}$ such that for any noticeable $\varepsilon(\lambda)$, any predicate $\phi(\cdot)$ and and any com^* such that $\phi\big(\mathsf{val}(\mathsf{com}^*)\big) = 0$,*[28] *it holds that*

$$\left\{(\mathsf{OUT}_{C^*},\eta_2) \;:\; (\mathsf{OUT}_{C^*},\eta_2) \leftarrow \langle C^*(\mathsf{ST}_{C^*}),R(\mathsf{com}^*)\rangle^{\phi}_{CF}\right\}_\lambda \overset{c}{\approx}_{\varepsilon} \left\{\big(\mathsf{OUT}_{C^*},\Gamma_d(\eta)\big) \;:\; \begin{array}{l}\eta \overset{\$}{\leftarrow} \{0,1\}^{k(\lambda)}\\ (\mathsf{OUT}_{C^*},d) \leftarrow \mathcal{S}(1^\lambda,\eta,\phi,\mathsf{com}^*)\end{array}\right\}_\lambda,$$

where $\Gamma_d(\eta)$ is defined as above.

Some remarks regarding Def. 24 follows:

- Note that the above completeness condition (Property 2) holds regardless of whether x satisfies the predicate $\phi(\cdot)$ or not.
- When defining Properties 3 and 4, we essentially follow the standard simulation-based notion for two-party coin-flipping *with aborting*. That is, we require the simulator to successfully 'enforce' the coin-flipping result to the pre-sampled η only if the malicious party does not abort the protocol. Indeed, if the malicious party aborts before the protocol ends, the honest party cannot receive any output, and thus of course we cannot enforce its output to η while being consistent with the real-world execution (where the honest party simply outputs $\bot$).

4.2 Construction

We present the construction in Prot. 2. It makes use of (the post-quantum version of) Naor's commitment and the post-quantum extractable commitment ExtCom with ε-simulation (as per Def. 3). Note that both of these building blocks are known in black-box and constant-round from post-quantum OWFs.

Protocol 2: Post-Quantum Trapdoor Coin-Flipping with ε-Simulation

Input: Both the C and R get the security parameter 1^λ as the common input. C gets a string $x \in \{0,1\}^{\ell(\lambda)}$ as its private input.

Commit Stage:

1. C prepares n views $\{\mathsf{v}_i\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string x (as detailed in Rmk. 5). C commits to each v_i ($i \in [n]$) independently in parallel, using Naor's commitment.

Decommit Stage:

1. C sends $\{\mathsf{v}_i\}_{i\in[n]}$ together with the decommitment information w.r.t. the commitments in Step 1.

[28] Recall from Def. 2 that $\mathsf{val}(\mathsf{com}^*)$ is the value statistically bound in transcript com^*.

2. R checks the validity of the decommitment information and the consistency among $\{\mathsf{v}_i\}_{i\in[n]}$ (as per Rmk. 3). If these checks are successful, R recovers x as $x \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{v}_1, \ldots, \mathsf{v}_n)$; otherwise, R rejects and output $\perp$.

Coin-Flipping Stage: At the beginning of this stage, both parties learn the description of a predicate $\phi(\cdot)$. They proceed as follows.

1. C picks $\eta_C \xleftarrow{\$} \{0,1\}^{k(\lambda)}$ and commits to it with MitH. Namely, C prepares n views $\{\mathsf{cfv}_i^{(1)}\}_{i\in[n]}$ corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the value η_C. C commits to each $\mathsf{cfv}_i^{(1)}$ $(i \in [n])$ independently in parallel, using ExtCom.
2. R picks $\eta_R \xleftarrow{\$} \{0,1\}^{k(\lambda)}$ and sends it to C.
3. C now sets $\eta'_C \coloneqq \eta_C$ and sends η'_C to R.
4. C prepares n views $\{\mathsf{cfv}_i^{(2)}\}_{i\in[n]}$, corresponding to an (n,k)-MitH execution of the n-party functionality F_ϕ described below, where party P_i $(i \in [n])$ uses $\mathsf{v}_i \| \mathsf{cfv}_i^{(1)}$ as input. (Note that $\mathsf{v}_i \| \mathsf{cfv}_i^{(1)}$ will be the prefix of $\mathsf{cfv}_i^{(2)}$.) C commits to each $\mathsf{cfv}_i^{(2)}$ $(i \in [n])$ independently in parallel, using Naor's commitment. Here, the honest committer will use as an 'effective witness' the value η_C reconstructed from $\{\mathsf{cfv}_i^{(2)}\}_{i\in[n]}$, and hence only evaluates the 'first clause' of F_ϕ (described below) in the virtual MPC execution.
   - **Functionality F_ϕ:** It has η'_C hare-wired. It collects input (and parses it as) $\mathsf{v}_i \| \mathsf{cfv}_i^{(1)}$ from party i for each $i \in [n]$. It then runs the recovery algorithm of VSS to obtain $a \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{v}_1, \ldots, \mathsf{v}_n)$ and $b \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cfv}_1^{(1)}, \ldots, \mathsf{cfv}_n^{(1)})$. It outputs 1 to each party if *either*
     - (First clause.) b equals η'_C sent in Step 3, *or*
     - (Second clause.) $\phi(a) = 1$.

     Otherwise, it outputs 0 to each party.
5. C and R now engage in the following coin-flipping subprotocol as detailed below.
   (a) R samples a random string θ_R of proper length and commits to it using ExtCom.
   (b) C samples a random string θ_C of proper length and sends it to R.
   (c) R sends to C the value θ_R together with the corresponding decommitment information w.r.t. the ExtCom in Step 5a. Now, C and R agree on a random string $\theta \coloneqq \theta_R \oplus \theta_C$. By a proper choice of length, the string θ it can be interpreted as specifying a size-k random subset of $[n]$. In the following, we abuse notation by using θ to denote the corresponding size-k random subset.
6. C sends back the list $\{(\mathsf{v}_i, \mathsf{cfv}_i^{(1)}, \mathsf{cfv}_i^{(2)})\}_{i\in\theta}$ together with the corresponding decommitment information w.r.t. the commitment made in Steps 1 and 4.
7. R now checks the validity of the decommitments provided by C and the consistency among the revealed views $\{(\mathsf{v}_i, \mathsf{cfv}_i^{(1)}, \mathsf{cfv}_i^{(2)}\}_{i\in\theta}$. It also checks for each $i \in \theta$ the final output of P_i contained in $\mathsf{cfv}_i^{(2)}$ is 1. If any of these checks fail, R aborts immediately.

Output: C and R output $\eta \coloneqq \eta'_C \oplus \eta_R$.

Security. The security of Prot. 2 is stated as the following lemma.

Lemma 7. *Prot. 2 is a black-box, constant-round trapdoor coin-flipping protocol (as per Def. 24).*

Proof. It is straightforward to see that Prot. 2 is constant-round and makes only black-box use of its underlying cryptographic components. Also, the completeness (i.e., Property 2) and statistically binding property (i.e., Property 1) of Prot. 2 is straightforward from the protocol description. Thus, to prove Lem. 7, we only need to show that Prot. 2 satisfies Properties 3 and 4 in Def. 24.

As we mentioned earlier, Prot. 2 is a simple application of the MPC-in-the-Head technique. Indeed, similar constructions have appeared previously in, e.g., [GLOV12, CCLY22a]. Our proof for Properties 3 and 4 follows almost immediately from these known techniques. In the following, we only provide a proof sketch and refer the interested readers to [CCLY22a, Section 6.5] for similar proofs.

Proving Property 3. Note that Prot. 2 can be viewed as a black-box commit-and-prove protocol where C commits to two values, one being the x committed in Step 1 of the **Commit Stage** and the other being η_C committed in Step 1 of the **Coin-Flipping Stage**; C then proves that these two values satisfies the predicate F_ϕ defined in Step 4. Note that such a black-box commit-and-prove protocol is *witness indistinguishable* (see [CCLY22a, Section 6.5] for a proof of this fact[29]), namely, even if C changes the committed values to another x' and η''_C, *as long as* x' *and* η''_C *together satisfy the predicate* F_ϕ, then no (potentially malicious) R^* could tell the difference.

With this observation, a simulator $\mathcal{S}$ can be constructed as follows. $\mathcal{S}$ commits to x' in Step 1 of the **Commit Stage** and emulates the honest C, where recall from the condition of Property 3 that x' is a valid witness for ϕ and is given to $\mathcal{S}$ as input. In Step 3, instead of setting η'_C to η_C honestly, $\mathcal{S}$ sets $\eta'_C = \eta \oplus \eta_R$, where recall that η is the input to $\mathcal{S}$ that it wants to enforce. Now, since $\eta'_C \neq \eta_C$, the 'first clause' of F_ϕ does not hold. Fortunately, since $\phi(x') = 1$, the 'second clause' of F_ϕ still holds. Thus, the new x' together with η_C constitute a valid witness for F_ϕ. Then, the witness indistinguishability of this commit-and-prove protocol implies Property 3.

Proving Property 4. This proof is even simpler. $\mathcal{S}$ works by extracting the η^*_C committed by C^* in Step 1 of the **Coin-Flipping Stage**, using the (parallel) extractability of ExtCom with ε-simulation. In more details, recall that the shares $\{\mathsf{cfv}_i^{(1)}\}_{i\in[n]}$ is committed in parallel using ExtCom in Step 1. $\mathcal{S}$ will extract all of these shares using the parallel extractability of ExtCom with ε-simulation (as per Def. 4), and compute $\eta^*_C := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cfv}_1^{(1)}, \ldots, \mathsf{cfv}_n^{(1)})$.

Then, $\mathcal{S}$ sends $\eta_R := \eta \oplus \eta^*_C$ in Step 2, and finishes the remaining execution emulating the honest R.

First, note that in a real-world execution between C^* and R, the value η^*_C (committed by C^* in Step 1 of the **Coin-Flipping Stage**) must equal to the η'_C sent by C^* in Step 3. This is because we know that the value committed by C^* in Step 1 of the **Commit Stage** does not satisfies the predicate $\phi(\cdot)$ (this is state as $\phi\big(\mathsf{val}(\mathsf{com}^*)\big) = 0$ in the condition of Property 4). Thus, if C^* sends $\eta'_C \neq \eta^*_C$ in Step 3, then both clauses of F_ϕ are unsatisfied. In this situation, C^* cannot provide a convincing proof to R; Otherwise, she breaks the soundness of the commit-and-prove protocol. Again, see [CCLY22a, Section 6.5] for a similar proof.

By the extractability of ExtCom, $\mathcal{S}$ is able to extract the same η^*_C while simulating the post-extraction state of C^* up to a arbitrarily small noticeable $\varepsilon(\lambda)$ error, and successfully enforce the coin-flipping result to $\eta^*_C \oplus \eta^*_C \oplus \eta = \eta$. This establishes Property 4.

[29] Indeed, [CCLY22a, Section 6.5] proves a stronger claim that such a protocol is ε-zero-knowledge, which implies witness indistinguishability.

This concludes the proof of Lem. 7.

□

5 Post-Quantum Non-Malleable Commitments: One-to-One and One-sided

We present a black-box, constant-round construction of post-quantum non-malleable commitment in the synchronous setting and supporting a polynomial number of tags. The construction shown in this section is non-malleable under the 'one-sided' assumption, i.e., its non-malleability holds in the 1-1 MIM execution where the left-session tag t is strictly smaller than the right-session tag $\widetilde{t}$. This constraint will be subsequently alleviated in Sec. 8.

5.1 Construction

The construction is described in Prot. 3. It makes black-box use of the following building blocks:

1. The post-quantum parallelly extractable commitment scheme ExtCom with ε-simulation Def. 4, which can be built in black-box from any post-quantum OWFs (see Lem. 3).
2. A post-quantum commitment scheme that is statistically-binding and computationally-hiding (against QPT adversaries). This is also known assuming only black-box access to post-quantum secure OWFs. In particular, we will make use of Naor's commitment which can be built in black-box from any post-quantum OWFs as well.
3. A perfectly secure verifiable secret sharing scheme $\mathsf{VSS} = (\mathsf{VSS}_{\mathsf{Share}}, \mathsf{VSS}_{\mathsf{Recon}})$ (as per Def. 8); We will set the parameters to make use of an $(n+1, k)$ scheme and an $(n+1, 2k)$ scheme. See Prot. 3 for details.
4. A semi-honest computationally private and perfectly robust MPC protocol (see Rmk. 4); We will set the parameters to make use of an (n, k) scheme and an $(n, 2k)$ scheme. See Prot. 3 for details.

We remark that the MPC and VSS are used to implement the MPC-in-the-Head (MitH) technique as explained in Sec. 3.7.

Protocol 3: One-Sided PQ-NMC: Black-Box and Constant-Round

Parameter Setting: The tag space is defined to be $[T]$, where T is a polynomial in the security parameter λ. Let n be a polynomial in λ, and k be a constant fraction of n such that $2k \leq n/3$.

Input: Both the committer C and the receiver R get the security parameter 1^λ and a tag $t \in [T]$ as the common input; C gets a string $m \in \{0,1\}^{\ell(\lambda)}$ as its private input, where $\ell(\cdot)$ is a polynomial.

Commit Phase:

1. **(Initial Com to m.)** In this stage, C commits to the message (using the MPC-in-the-head approach).
   - C prepares n views $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1, 2k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the message m (as detailed in Rmk. 5). C commits to each $\mathsf{cv}_i^{(1)}$ $(i \in [n])$ independently in parallel, using Naor's commitment.
2. **(Hard Puzzle Setup.)** In this stage, R sets up a t-solution hard puzzle. It then commits to one solution of the puzzle and proves in zero-knowledge the consistency (using the MPC-in-the-head approach).

(a) C samples a size-k random subset $\mathsf{ch} \subseteq [n]$, and commits to it using ExtCom.

(b) R samples t random strings $x_1, \ldots, x_t \xleftarrow{\$} \{0,1\}^\lambda$. R prepares n views $\{\mathsf{rv}_i^{(1)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string $x_1\|\ldots\|x_t$ (as detailed in Rmk. 5). R commits to each $\mathsf{rv}_i^{(1)}$ $(i \in [n])$ independently in parallel, using Naor's commitment.

(c) R prepares another n views $\{\mathsf{rv}_i^{(2)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string $1\|x_1$ (as detailed in Rmk. 5). R commits to each $\mathsf{rv}_i^{(2)}$ $(i \in [n])$ independently in parallel, using ExtCom.

(d) R then prepares another n views $\{\mathsf{rv}_i^{(3)}\}_{i\in[n]}$, corresponding to an (n,k)-MitH execution of the n-party functionality F^R_{consis} described below (intuitively, F^R_{consis} checks the consistency between Step 2b and Step 2c), where party P_i $(i \in [n])$ uses $\mathsf{rv}_i^{(1)}\|\mathsf{rv}_i^{(2)}$ as input. (Note that $\mathsf{rv}_i^{(1)}\|\mathsf{rv}_i^{(2)}$ will be a prefix of $\mathsf{rv}_i^{(3)}$.) R commits to each $\mathsf{rv}_i^{(3)}$ $(i \in [n])$ independently in parallel, using Naor's commitment.

- **Functionality F^R_{consis}:** It collects input (and parses it as) $\mathsf{rv}_i^{(1)}\|\mathsf{rv}_i^{(2)}$ from party i for each $i \in [n]$. It then runs the recovery algorithm of VSS to obtain $a_1\|\ldots\|a_t := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{rv}_1^{(1)}, \ldots, \mathsf{rv}_n^{(1)})$ and $j\|b_j := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{rv}_1^{(2)}, \ldots, \mathsf{rv}_n^{(2)})$. If $j \in [t]$ and $b_j = a_j$, it outputs 1 to each party; otherwise, it outputs 0 to each party.

(e) C sends ch together with the decommitment information (w.r.t. Step 2a).

(f) R sends $\{(\mathsf{rv}_i^{(1)}, \mathsf{rv}_i^{(2)}, \mathsf{rv}_i^{(3)})\}_{i\in\mathsf{ch}}$ together with the decommitment information (w.r.t. their respective commitments in Steps 2b to 2d).

(g) C checks the validity of the decommitment information and the consistency among the revealed views $\{(\mathsf{rv}_i^{(1)}, \mathsf{rv}_i^{(2)}, \mathsf{rv}_i^{(3)})\}_{i\in\mathsf{ch}}$. In particular, it checks for each $i \in \mathsf{ch}$ that $\mathsf{rv}_i^{(1)}\|\mathsf{rv}_i^{(2)}$ is the prefix of $\mathsf{rv}_i^{(3)}$. It also checks for each distinct pair $i, j \in \mathsf{ch}$ that $(\mathsf{rv}_i^{(1)}, \mathsf{rv}_j^{(1)})$ are consistent, $(\mathsf{rv}_i^{(2)}, \mathsf{rv}_j^{(2)})$ are consistent, and $(\mathsf{rv}_i^{(3)}, \mathsf{rv}_j^{(3)})$ are consistent, where by 'consistent' we refer to the consistency requirements as per Def. 10 and Rmk. 3. It also checks for each $i \in \mathsf{ch}$ the final output of P_i contained in $\mathsf{rv}_i^{(3)}$ is 1. It aborts immediately if any of the checks fail.

3. **(ExtCom to m.)** C commits to m once again, using ExtCom in the MitH format. In more detail:

   - C prepares n views $\{\mathsf{cv}_i^{(2)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,2k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the message m (as detailed in Rmk. 5). C commits to each $\mathsf{cv}_i^{(2)}$ $(i \in [n])$ independently in parallel, using ExtCom.

   We remark that this step can be viewed as the **Commit Stage** of the trapdoor coin-flipping protocol shown in Prot. 2. Here are two key points we want to emphasize: (1) The original Prot. 2 uses Naor's commitment for this stage, but here we instead use ExtCom. This is to make this commitment extractable. (2) The original Prot. 2 uses a $(n+1,k)$-VSS scheme, but here we instead use a $(n+1,2k)$ VSS scheme. That is because we need to open two subsets of size k, one in Step 5b, which is the **Coin-Flipping Stage**, and the other in Step 5c, which is for the proof of consistency of the current PQ-NMC protocol.
   These modifications will be important later when we prove non-malleability.

4. **(Puzzle Solution Reveal.)** R reveals $(x_1, \ldots, x_t)$ by decommitting to $\{\mathsf{rv}_i^{(1)}\}_{i\in[n]}$. In more detail,

   (a) R sends $\{\mathsf{rv}_i^{(1)}\}_{i\in[n]}$ together with the decommitment information w.r.t. the commitments in Step 2b.

   (b) C checks the validity of the decommitment information and the consistency among $\{\mathsf{rv}_i^{(1)}\}_{i\in[n]}$ (as per Rmk. 3). If these checks are successful, C recovers $x_1\|\ldots\|x_t \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{rv}_1^{(1)}, \ldots, \mathsf{rv}_n^{(1)})$; otherwise, C rejects and output $\perp$.

5. **(Committer's Consistency Proof.)** This stage should be interpreted as C proving consistency between its actions in Steps 1 and 3 (i.e., these two steps commit to the same value) using a witness indistinguishable argument, where the trapdoor statement is that C manages to commit to a puzzle solution in Step 3. This step is again conducted in the MitH format. Note that for the honest committer, the 'effective witness' in MitH is the message m reconstructed from both $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$ and $\{\mathsf{cv}_i^{(2)}\}_{i\in[n]}$, and so the virtual MPC execution in reality evaluates the 'first clause' of F_{consis}^C as defined below. In more detail, this stage proceeds as follows.

   (a) C prepares n views $\{\mathsf{cv}_i^{(3)}\}_{i\in[n]}$, corresponding to an $(n, 2k)$-MitH execution of the n-party functionality F_{consis}^C described below, where party P_i ($i \in [n]$) uses $\mathsf{cv}_i^{(1)}\|\mathsf{cv}_i^{(2)}$ as input. (Note that $\mathsf{cv}_i^{(1)}\|\mathsf{cv}_i^{(2)}$ will be the prefix of $\mathsf{cv}_i^{(3)}$.) C commits to each $\mathsf{cv}_i^{(3)}$ ($i \in [n]$) independently in parallel, using Naor's commitment.

   - **Functionality F_{consis}^C:** It collects input (and parses it as) $\mathsf{cv}_i^{(1)}\|\mathsf{cv}_i^{(2)}$ from party i for each $i \in [n]$. It then runs the recovery algorithm of VSS to obtain $a \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cv}_1^{(1)}, \ldots, \mathsf{cv}_n^{(1)})$ and $b \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cv}_1^{(2)}, \ldots, \mathsf{cv}_n^{(2)})$. It outputs 1 to each party if *either*
     - (First clause.) $b = a$, *or*
     - (Second clause.) b can be parsed as $j\|x'$ such that $j \in [t]$ and $x' = x_j$ (recall that x_j is among the puzzle solutions revealed by R in Step 4.

     Otherwise, it outputs 0 to each party.

   (b) **(Trapdoor Coin-Flipping)** C and R then execute the **Coin-Flipping Stage** of the trapdoor coin-flipping protocol shown in Prot. 2, with the trapdoor predicate $\phi(\cdot)$ defined as follows

   - **Predicate $\phi(\cdot)$:** It has the values $(x_1, \ldots, x_t)$ hard-wired (recall that these values are revealed in Step 4). On input $i\|a$, it outputs 1 iff $i \in [t]$ and $a = x_i$.

   By the completeness of the trapdoor coin-flipping protocol (i.e., Property 2 in Def. 24), at the end of this step, C and R agree on a string η. By a proper choice of length, the string η can be interpreted as specifying a size-k random subset of $[n]$. In the following, we abuse notation by using η to denote the corresponding size-k random subset.

   (c) C sends $\{(\mathsf{cv}_i^{(1)}, \mathsf{cv}_i^{(2)}, \mathsf{cv}_i^{(3)})\}_{i\in\eta}$ together with the decommitment information (w.r.t. their respective commitments in Steps 1, 3 and 5a).

   (d) R checks the validity of the decommitment information and the consistency among the revealed views $\{(\mathsf{cv}_i^{(1)}, \mathsf{cv}_i^{(2)}, \mathsf{cv}_i^{(3)})\}_{i\in\eta}$. It also checks for each $i \in \eta$ the final output of P_i contained in $\mathsf{cv}_i^{(3)}$ is 1. R aborts if any of these checks fail.

Decommit Stage:

1. C sends $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$ together with the decommitment information w.r.t. the commitments in Step 1.
2. R checks the validity of the decommitment information and the consistency among $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$ (as per Rmk. 3). If these checks are successful, R recovers m as $m \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cv}_1^{(1)}, \ldots, \mathsf{cv}_n^{(1)})$; otherwise, R rejects and output $\perp$.

Security. The security of Prot. 3 is stated as the following theorem.

Theorem 6. *Assuming the existence of post-quantum one-way functions, there exists (i.e., Prot. 3) a black-box, constant-round construction of 1-1 post-quantum non-malleable commitments (as per Def. 5 with $k = 1$) in synchronous setting, with one-sided security, and supporting tag space $[T]$ with $T(\lambda)$ being any polynomial in the security parameter λ.*

It is straightforward to see that Prot. 3 is constant-round and makes only black-box use of its underlying cryptographic components. Completeness of Prot. 3 is also straightforward from the protocol description. The statistical binding property follows from that of Naor's commitment in Step 1. Computational-hiding property of any non-malleable commitment scheme follows directly from its non-malleability. So, to prove Thm. 6, we only need to prove the post-quantum non-malleability of Prot. 3, which we prove in subsequent subsections.

5.2 Outline for the Proof of Non-Malleability

The proof for the non-malleability of Prot. 3 is very involved and lengthy. To help the reader understand it better, we provide an outline delving into it.

Recall from Def. 5 (with $k = 1$) that to prove non-malleability of Prot. 3, we consider the synchronous 1-1 MIM execution of Prot. 3 where the left session uses tag t and the right session uses tag $\widetilde{t}$. Also recall that $\widetilde{t} \geq t+1$ since we focus on the 'one-sided' setting. We need to show that for any QPT MIM adversary $\mathcal{M}_\lambda(\rho_\lambda)$, it holds that

$$\left\{\mathsf{mim}^{\mathcal{M}_\lambda}(m_0, \rho_\lambda)\right\}_{\lambda\in\mathbb{N}, m_0, m_1\in\{0,1\}^{\ell(\lambda)}} \overset{\mathrm{c}}{\approx} \left\{\mathsf{mim}^{\mathcal{M}_\lambda}(m_1, \rho_\lambda)\right\}_{\lambda\in\mathbb{N}, m_0, m_1\in\{0,1\}^{\ell(\lambda)}}, \tag{4}$$

where $\mathsf{mim}^{\mathcal{M}_\lambda}(m_b, \rho_\lambda)$ denotes the *joint* distribution of the output of $\mathcal{M}_\lambda$ *and* the value $\widetilde{m}$ committed in the right session, where the left (honest) committer C commits to message m_b.

At a high level, our proof follows the template from [LPY23a]. We reduce non-malleability to the computational-hiding property of the Naor's commitment in Step 1 of the left session. That is, we consider in the MIM execution that the Naor's commitment in Step 1 of the left session comes from an external challenger, committing to an underlying message m_b with b picked uniformly from $\{0,1\}$. Our goal is to construct a machine $\mathcal{SE}$ (dubbed simulation-extractor) that can efficiently extract the value $\widetilde{m}$ committed by $\mathcal{M}_\lambda$ in the left session, while simulating $\mathcal{M}_\lambda$'s post-extraction state (possibly with an arbitrarily small simulation error ε). Note that if such an $\mathcal{SE}$ exists, it indeed *efficiently* outputs the value $\mathsf{mim}^{\mathcal{M}_\lambda}(m_b, \rho_\lambda)$ (which was not efficiently computable since the $\widetilde{m}$ value is hidden in the transcript). Then, if Eq. (4) does not hold, $\mathcal{SE}$'s output will be distinguishable when the external challenger changes the committed value between m_0 and m_1. This breaks the computational hiding property of Naor's commitment.

However, there are some challenges in implementing this template.

1. First, note that the message m committed in Step 1 of the left session is used again in Step 3 of the left session. Thus, to be able to forward the left Step 1 to the external challenger, we have to come up with some method to avoid using the underlying m in Step 3 of the left session (as this m is known only to the external challenger in the reduction we outlined above).
2. In sharp contrast to the original [LPY23a] construction, Step 1 of our Prot. 3 is not a Naor's commitment to the underlying message m. Rather, it is a parallel execution of Naor's scheme committing to n VSS shares of the message m. Moreover, note that Steps 1, 3 and 5 of Prot. 3 essentially consist of a black-box commit-and-prove protocol, where C first commits to two messages in Steps 1 and 3 respectively, and then proves a predicate F^{C}_{consis} over the two committed values. This structure induces new challenges: Even if we can make Step 3 independent of the m committed in Step 1, it still need to reveal some of the shares committed in Step 1 when proving consistency in Step 5 (in particular, in Step 5c).

Resolving these issues requires us to use new ideas described in Sec. 2. In more detail, we will first show that *assume the existence of the* $\mathcal{SE}$, we can still follow the same template but rather reduce non-malleability to the VSS hiding game shown in Expr. 1, instead of to the vanilla Naor's commitment. We then show that the desired $\mathcal{SE}$ can indeed be constructed. In the following, we present an outline for these two steps.

Reducing Non-Malleability to VSS Hiding. This step is performed in Sec. 5.3. It is organized as follows.

1. We the first 'decouple' the the committed message in Step 1 of the left session from the remaining steps. To do that, we design a game $\widetilde{H}^{\mathcal{M}_\lambda}$ in Algo. 5.2. This $\widetilde{H}^{\mathcal{M}_\lambda}$ is essentially identical to the 1-1 MIM execution, but makes use of a new machine $\mathcal{G}_1$ (in Algo. 5.3) to finish the steps after Step 1. A key feature of $\mathcal{G}_1$ is that it does not need to know the m committed in Step 1 of the left session.
2. Next, we define another game $\widetilde{G}^{\mathcal{M}_\lambda}$ (in Algo. 5.4), which performs Step 1 in the same manner as $\widetilde{H}^{\mathcal{M}_\lambda}$, but replace the $\mathcal{G}_1$ machine by a simulation extractor $\mathcal{SE}$ *that we assume to exist*. We will show (in Lem. 9) that this $\mathcal{SE}$ can extract from $\mathcal{G}_1$ the $\widetilde{m}$ committed by $\mathcal{M}_\lambda$ in the right session, while simulating the post-extraction state of $\mathcal{G}_1$ (with a noticeable error ε that can be made arbitrarily small).
3. Finally, we show that the machine $\widetilde{G}^{\mathcal{M}_\lambda}$ is designed on purpose so that we can reduce non-malleability to the VSS hiding game. This is done in Sec. 5.4.

Building Simulation-Extractor $\mathcal{SE}$. To build the desired $\mathcal{SE}$, we first build a machine $\mathcal{K}$ that is able to extract the correct $\widetilde{m}$ from the machine $\mathcal{G}_1$ mentioned above. But $\mathcal{K}$ is not capable of simulating the post-extraction state of $\mathcal{G}_1$. We will also show that $\mathcal{K}$ satisfies some extra requirements so that we can later convert it to an extractor *with simulation*. The description of $\mathcal{K}$ and the proof for its properties are the focus of Sec. 6.

We then show how to equip $\mathcal{K}$ with simulation in Sec. 7. To do that, we first need a generalization of the counterpart lemma from [LPY23b, Lemma 20]. This is because our construction makes heavy use of black-box commit-and-prove techniques so that the simulation-less extractor $\mathcal{K}$ satisfies only weaker properties than its counterpart in [LPY23b, Lemma 31]. In particular, our $\mathcal{K}$ cannot check if the value it extracted is indeed the correct $\widetilde{m}$. Thus, we need to generalize the simulation-extraction lemma in [LPY23b, Lemma 20] to take care of related issues. This is handled in Sec. 7.1.

Finally, with all the preparatory work before, we can eventually convert our $\mathcal{K}$ to the desired $\mathcal{SE}$ using the a 'noisy' simulation-extraction lemma developed in Sec. 7.1. This is done in Sec. 7.2.

This finishes the outline for our proof of non-malleability.

5.3 Reduction to VSS Hiding Game

We first define some notion related to the MIM execution of Prot. 3.

Algorithm 5.1: Machine $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$

Machine $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$**:** This is the man-in-the-middle execution of the commit stage of Prot. 3, where the left committer commits to m and $\mathcal{M}_\lambda$'s non-uniform advice is ρ_λ. The output of this game is denoted by $\mathsf{Out}_{H^{\mathcal{M}_\lambda}}(\lambda, m, \rho_\lambda)$ and consists of the following three parts:

1. OUT: This is the (quantum) output of $\mathcal{M}$ at the end of this game;
2. $\widetilde{\tau}$: This is the commitment transcript sent by $\mathcal{M}$ in the Step 1 of the right session;
3. $\widetilde{d} \in \{\top, \bot\}$: This is the output of the honest receiver R in the right session, indicating if the man-in-the-middle's commitment (i.e., the right session) is accepted ($\widetilde{d} = \top$) or not ($\widetilde{d} = \bot$).

Also, to prove non-malleability, we need to talk about the value committed in the right session. Toward that, we define the following function:

$$\mathsf{val}_{\widetilde{d}}(\widetilde{\tau}) := \begin{cases} \mathsf{val}(\widetilde{\tau}) & \widetilde{d} = \top \\ \bot & \widetilde{d} = \bot \end{cases},$$

where $\mathsf{val}(\widetilde{\tau})$ denote the value statistically-bound $\widetilde{\tau}$ (i.e., the value that can be re-constructable from $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$). Note that $\mathsf{val}_{\widetilde{d}}(\widetilde{\tau})$ is exactly the value committed in the right session by $\mathcal{M}$. Thus, to prove satisfies Def. 5 (when $k = 1$), we only need to establish the following equation:

$$\begin{aligned} &\left\{\left(\mathsf{OUT}^0, \mathsf{val}_{\widetilde{d}^0}(\widetilde{\tau}^0)\right) : (\mathsf{OUT}^0, \widetilde{\tau}^0, \widetilde{d}^0) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m_0, \rho_\lambda)\right\} \\ \overset{\mathrm{c}}{\approx} &\left\{\left(\mathsf{OUT}^1, \mathsf{val}_{\widetilde{d}^1}(\widetilde{\tau}^1)\right) : (\mathsf{OUT}^1, \widetilde{\tau}^1, \widetilde{d}^1) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m_1, \rho_\lambda)\right\}, \end{aligned} \tag{5}$$

where both ensembles are indexed by $\lambda \in \mathbb{N}$ and $(m_0, m_1) \in \{0,1\}^{\ell(\lambda)} \times \{0,1\}^{\ell(\lambda)}$.

Next, we describe a new game $\widetilde{H}^{\mathcal{M}_\lambda}$, which is a slight modification of the game $\widetilde{H}^{\mathcal{M}_\lambda}$. It will help us switch out different commitments in the left interaction.

Algorithm 5.2: Game $\widetilde{H}^{\mathcal{M}_\lambda}(\lambda, \varepsilon, m, \rho_\lambda)$

Input: It takes as input the same parameters λ, ρ_λ, and m as for $H^{\mathcal{M}_\lambda}$. It additionally takes as input a noticeable function $\lambda(\cdot)$.
It proceeds as follows:

1. **(Prefix phase.)** This proceeds as follows.
   (a) Sample a random size-k subset $\eta \subset [n]$.
   (b) Execute $H^{\mathcal{M}_\lambda}(m_1, \lambda, \rho_\lambda)$ until the end of Step 1. At the moment, it already receives the Step 1 commitment made by the left-session honest committer C. It performs brute-force computation to obtain from C's commitment the committed shares cv_i and their decommitment information for $i \in \eta$. We denote these values as $\mathsf{VI}_\eta := \{(\mathsf{cv}_i, \mathsf{decom}_i\}_{i\in\eta}$.

   Notation: Let $\mathsf{st}_{\mathcal{M}}$ denote the state of $\mathcal{M}$ at the end of Step 1; Let st_C (resp. st_R) denote the state of the honest committer (resp. receiver) at the end of Step 1; Let $\widetilde{\tau}$ denote the commitment sent by $\mathcal{M}$ in Step 1 of the right session. We denote the tuple $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ as pref. We use

the following nation to express the execution of this **Prefix phase**:

$$(\mathsf{pref}, \eta, \mathsf{VI}_\eta) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pref}}(\lambda, m, \rho_\lambda). \tag{6}$$

We will often use $\mathsf{pref}' := (\mathsf{pref}, \eta, \mathsf{VI}_\eta)$ to refer to the concatenation of pref and the η, VI_η. We remark that this prefix generation step is independent of the error parameter ε.

2. **(Remainder phase.)** This involves the following steps:
   (a) $\widetilde{H}^{\mathcal{M}_\lambda}$ now invokes $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta))$ (as described in Algo. 5.3), which outputs a tuple $(\mathsf{OUT}, \widetilde{d})$.
   (b) $\widetilde{H}^{\mathcal{M}_\lambda}$ outputs $(\mathsf{OUT}, \widetilde{\tau}, \widetilde{d})$.

We now describe the subprocedure $\mathcal{G}_1(\cdot)$.

Algorithm 5.3: Machine $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)$

Game $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)$ continues the execution using pref just as in $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$, apart from the following differences:

1. In the left Step 2c against $\mathcal{M}_\lambda$, instead of following the honest receiver algorithm for ExtCom, it instead uses the extractor $\mathcal{SE}^{\mathcal{M}_\lambda}_{\mathsf{ExtCom}}(1^\lambda, 1^{\varepsilon^{-1}})$ to obtain an extracted value of the form $j'||x'_{j'}$ (and continuing the execution with the simulated state).
   In more detail, the shares $\{\mathsf{rv}_i^{(2)}\}_{i\in[n]}$ are committed by $\mathcal{M}_\lambda$ using independent ExtCom in parallel. $\mathcal{G}_1$ will extract all of these shares using the parallel extractability of ExtCom with ε-simulation (as per Def. 4), and compute $j'||x'_{j'} := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{rv}_1^{(2)}, \ldots, \mathsf{rv}_n^{(2)})$.
2. In the ExtCom execution for Step 3 on the left, it commits to the extracted value $j'\|x'_{j'}$.
   In more detail, $\mathcal{G}_1$ first prepares n views $\{\mathsf{cv}_i^{(2)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1, 2k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the value $j'\|x'_{j'}$ extracted in Step 2, and then commits to each $\mathsf{cv}_i^{(2)}$ $(i \in [n])$ independently in parallel, using ExtCom.
3. In Step 4 of the left session, after reconstructing the receiver's puzzle solutions $x_1\| \ldots \|x_t$, it checks whether $x_{j'} = x'_{j'}$, i.e., the value that it extracted earlier. If not, it aborts the execution and outputs $\perp$.
4. In Step 5a of the left session, instead of generating the views $\{\mathsf{cv}_i^{(3)}\}_{i\in[n]}$ using the 'first clause' of F^C_{consist}, generate these views using the 'second clause'. We remark that this is possible because in Step 2 above, we already commit to the extracted $j'\|x'_{j'}$, which satisfies the 'second clause' of F^C_{consist}.
   Recall that each $\mathsf{cv}_i^{(3)}$ has $\mathsf{cv}_i^{(1)}$ as a prefix. However, $\mathcal{G}_1$ only knows the $\mathsf{cv}_i^{(1)}$ for $i \in \eta$. For that, $\mathcal{G}_1$ simply set $\mathsf{cv}_i^{(1)}$ for $i \in [n] \setminus \eta$ to all-0 strings. This does not affect this step as $\mathcal{G}_1$ now is proving the 'second clause' of F^C_{consist}, which is independent of the value determined by the real $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$. (Also note that this problem does not occur for $\{\mathsf{cv}_i^{(2)}\}_{i\in[n]}$, which is generated by $\mathcal{G}_1$ itself in Step 2.)
5. In Step 5b of the left session, instead of executing the trapdoor coin-flipping protocol honestly, use the 'straight-line' simulator $\mathcal{S}(\mathsf{ST}_C, \eta, j'\|x'_{j'}, \phi)$ that is guaranteed to exist by Property 3 in Def. 24 (where ST_C is the classical state of the committer at the end of Step 3, emulated by $\mathcal{G}_1$). Note that this is possible because we current have $\phi(j'\|x'_{j'}) = 1$. This effectively 'enforce' the coin-flipping result to the η contained in the input to $\mathcal{S}$.

6. It concludes the execution by performing the remaining steps as in $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$. One caveat is: In Step 5c of the left session, $\mathcal{G}_1$ will be asked to decommitment to $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]\setminus\eta}$. Note that the η is already 'enforced' to the η in Step 5, and $\mathcal{G}_1$ does know the $\mathsf{cv}_i^{(i)}$ shares and decommitment information for $i \in \eta$ (contained in VI_η).
7. It finally outputs the values $(\mathsf{OUT}, \widetilde{d})$, where again OUT is $\mathcal{M}$'s final output and $\widetilde{d}$ is the honest R's final decision in the right session.

Lemma 8. *For all $m \in \{0,1\}^{\ell(\lambda)}$ and all noticeable $\varepsilon(\cdot)$, it holds that*

$$\left\{\left(\mathsf{OUT}, \mathsf{val}_{\widetilde{d}}(\widetilde{\tau})\right) : (\mathsf{OUT}, \widetilde{\tau}, \widetilde{d}) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)\right\}_\lambda \overset{c}{\approx}_\varepsilon \left\{\left(\mathsf{OUT}, \mathsf{val}_{\widetilde{d}}(\widetilde{\tau})\right) : (\mathsf{OUT}, \widetilde{\tau}, \widetilde{d}) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}(\lambda, \varepsilon, m, \rho_\lambda)\right\}_\lambda.$$

Proof. We prove this lemma by a hybrid argument. Some of the modifications are pretty standard within black-box MPC literature and we sketch them for brevity. In the following, we fix arbitrary an m and an ε. For each hybrid H_i, we use OUT_{H_i} to denote its output.

Hybrid H_0: This is simply the game $H^{\mathcal{M}_\lambda}(m, \rho_\lambda)$, renamed for convenience.

Hybrid H_1: This hybrid is identical to the previous one, except for the following changes. In Step 2c of the left session, instead of following the honest C's algorithm, it uses the extractor $\mathcal{SE}^{\mathcal{M}_\lambda}_{\mathsf{ExtCom}}(1^\lambda, 1^{\varepsilon^{-1}})$ to obtain an extracted a value of the format $j'\|x'_{j'}$ and a (simulated) state $\mathsf{st}'_{\mathcal{M}}$ (as described in Step 2 of $\mathcal{G}_1$). It records $j'\|x'_{j'}$ and continues the execution of the MIM interaction with $\mathsf{st}'_{\mathcal{M}}$ up till Step 4, where it obtains $x_1, \ldots, x_t$ and checks if $x_{j'} = x'_{j'}$. If not, it aborts; otherwise it finishes the execution. All other steps are carried out as in the previous hybrid. Note that H_1 now requires the additional input $1^{\varepsilon^{-1}}$.

$\mathsf{Out}_{H_0} \overset{s}{\approx}_\varepsilon \mathsf{Out}_{H_1}$: This follows directly from the parallel extractability with ε-simulation of ExtCom (as per Def. 4).

Hybrid H_2: This hybrid is identical to the previous one, except for the following changes. In Step 3 of the left session, it uses the extracted $j'\|x'_{j'}$ as the committed message; And in Step 5a of the left session, instead of generating the views $\{\mathsf{cv}_i^{(3)}\}_{i\in[n]}$ using the 'first clause' of F^C_{consist}, generate these views using the 'second clause'.

Remark 6. We remark that this step is slight different from Step 4 of $\mathcal{G}_1$, where the shares $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]\setminus\eta}$ (as prefix of $\mathsf{cv}_i^{(3)}$'s) is set to 0-strings. In the current hybrid, H_2 still uses the honest $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]\setminus\eta}$ shares, because these shares are generated by itself and thus it knows them. We will change these shares to 0-strings in a later hybrid (i.e., H_4).

$\mathsf{Out}_{H_1} \overset{c}{\approx} \mathsf{Out}_{H_2}$: Note that Step 3 and Step 5 constitutes a black-box commit-and-prove protocol. What we did in this step is to switch from one witness for the target predicate to another witness. This switch is computationally indistinguishable. Since this argument is standard, we omit the details (see [CCLY22a, Section 6.5] for an example).

Hybrid H_3: This hybrid is identical to the previous one, except for the following changes. At the very beginning, it samples a random size-k subset $\eta \subset [n]$; and In Step 5b of the left session, instead of executing the trapdoor coin-flipping protocol honestly, it uses the 'straight-line' simulator $\mathcal{S}(\mathsf{ST}_C, \eta, j'\|x'_{j'}, \phi)$ to 'enforce' the coin-flipping result (in the same manner as described in Step 5 of $\mathcal{G}_1$).

$\underline{\mathsf{Out}_{H_2} \overset{c}{\approx} \mathsf{Out}_{H_3}}$: This follows directly from the security guarantee of $\mathcal{S}$ (i.e., Property 3 in Def. 24).

Hybrid H_4: This hybrid is identical to the previous one, except for the following changes. In Step 5a of the left session, when generating $\mathsf{cv}_i^{(3)}$, it sets the shares $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]\setminus\eta}$ to 0-strings. This is to compensate the concern in Rmk. 6.

$\underline{\mathsf{Out}_{H_3} \overset{c}{\approx} \mathsf{Out}_{H_4}}$: First, note that changing the shares $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]\setminus\eta}$ does not affect Step 5a of the left session, as we already switch to using the witness for the 'second clause' of it in H_3. Also note that these shares will never be revealed because in H_2 we already 'enforce' the challenge set to η. Thus, the indistinguishability of H_3 and H_4 follows directly from the computational hiding property of Naor's commitment in Step 5a of the left session.

Hybrid H_5: Note that in H_4, we already do not need to use any information about the shares $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]\setminus\eta}$. In this hybrid, we can think that after Step 1 of the execution, H_5 performs brute-force computation to learn the shares $\{\mathsf{cv}_i^{(1)}\}_{i\in\eta}$ and their corresponding decommitment information (and put them together as VI_η). Indeed, these are the only information that is need to finish the remaining execution.

$\underline{\mathsf{Out}_{H_4} \overset{\text{i.d.}}{=\!=} \mathsf{Out}_{H_5}}$: There is no real change in H_5 other than a change of perspective. These two hybrids are thus identical.

Finally, note that H_5 is exactly $\widetilde{H}^{\mathcal{M}_\lambda}(\lambda, \varepsilon, m, \rho_\lambda)$. This concludes the proof of Lem. 8.

□

Next, we define a further modified game $\widetilde{G}$ that instead invokes a particular simulator-extractor $\mathcal{SE}$ that helps obtain the value committed to by $\mathcal{M}_\lambda$ on the right (which then helps to demonstrate non-malleability).

Algorithm 5.4: Game $\widetilde{G}^{\mathcal{M}_\lambda}(\lambda, \varepsilon, m, \rho_\lambda)$

This proceeds in two phases as well:

1. **(Prefix phase.)** This is identical to the prefix phase of Algo. 5.2, i.e., it computes $(\mathsf{pref}, \eta, \mathsf{VI}_\eta) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pref}}(\lambda, m, \rho_\lambda)$.
2. **Remainder phase:** This involves the following steps:
   - It invokes a machine $\mathcal{SE}$, which is guranteed to exist by the following Lem. 9: $\mathcal{SE}$ takes in as input a tuple $(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)$ and outputs $(\mathsf{OUT}, \mathsf{Val})$.
   - $\widetilde{G}^{\mathcal{M}_\lambda}$ outputs $(\mathsf{OUT}, \mathsf{Val})$ as its own output.

The following Lem. 9 serves as assurance that we can build such a machine $\mathcal{SE}$ so that the games $\widetilde{H}^{\mathcal{M}_\lambda}$ and $\widetilde{G}^{\mathcal{M}_\lambda}$ present ε-close views to the adversary (where we control the closeness parameter). Lem. 9 represents the most challenging task in the current proof of non-malleability. We will prove it in Sec. 6 and 7

Lemma 9 (1-1 Simulation-Extractor). *Let $\mathcal{G}_1(\cdot)$ be the efficient procedure defined in Algo. 5.3. There exists a simulation-extractor $\mathcal{SE}$ such that for any $(\mathsf{pref}, \eta, \mathsf{VI}_\eta)$ in the support of $\widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pre}}$, and for any noticeable $\varepsilon(\lambda)$, there is a noticeable $\varepsilon'(\lambda) \leq 8\varepsilon(\lambda)$ that is efficiently computable form $\varepsilon(\lambda)$ such that the following holds:*

$$\begin{aligned}&\left\{(\mathsf{OUT}, \mathsf{Val}) : (\mathsf{OUT}, \mathsf{Val}) \leftarrow \mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)\right\}\\ \overset{c}{\approx}_\varepsilon\ &\left\{\left(\mathsf{OUT}, \mathsf{val}_{\widetilde{d}}(\widetilde{\tau})\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon'^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)\right\}.\end{aligned}$$

The following Corollary 4 is an immediate consequence of Lem. 9.

Corollary 4. *Let $\widetilde{H}^{\mathcal{M}_\lambda}$ and $\widetilde{G}^{\mathcal{M}_\lambda}$ be as defined in Algo. 5.2 and Algo. 5.4 respectively. For any QPT adversary $\mathcal{M}_\lambda(\rho_\lambda)$, any $m \in \{0,1\}^{\ell(\lambda)}$, and any noticeable $\varepsilon(\lambda)$, there is a noticeable $\varepsilon'(\lambda) \leq 8\varepsilon(\lambda)$ that is efficiently computable form $\varepsilon(\lambda)$ such that the following holds:*

$$\begin{aligned}&\left\{(\mathsf{OUT},\mathsf{Val}) : (\mathsf{OUT},\mathsf{Val}) \leftarrow \widetilde{G}^{\mathcal{M}_\lambda}(\lambda,\varepsilon,m,\rho_\lambda)\right\}\\ \overset{c}{\approx}_\varepsilon &\left\{\left(\mathsf{OUT},\mathsf{val}_{\widetilde{d}}(\widetilde{\tau})\right) : (\mathsf{OUT},\widetilde{\tau},\widetilde{d}) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}(\lambda,\varepsilon',m,\rho_\lambda)\right\}.\end{aligned}$$

Proof. Note that the prefix stages of $\widetilde{H}^{\mathcal{M}_\lambda}$ (see Step 1 in Algo. 5.2) and $\widetilde{G}^{\mathcal{M}_\lambda}$ (see Step 1 in Algo. 5.4) are identical, and therefore Lem. 9 immediately applies to show that $(\mathsf{OUT},\mathsf{val}_{\widetilde{d}}(\widetilde{\tau}))$ obtained by running $\widetilde{H}^{\mathcal{M}_\lambda}$ and $(\mathsf{OUT},\mathsf{Val})$ obtained by running $\widetilde{G}^{\mathcal{M}_\lambda}$ are ε-close.

□

5.4 Finishing the Proof of Non-Malleability

With the helper machines defined in Sec. 5.3, we can now finish the proof of non-malleability. This is a proof by contradiction. We will show that if the non-malleability of Prot. 3 does not hold, then the machine $\widetilde{G}^{\mathcal{M}_\lambda}$ can be used to break the VSS hiding game defined in Expr. 1. We present the formal argument in the following.

We first show a lemma that relates machine $\widetilde{G}^{\mathcal{M}_\lambda}$ to the VSS hiding game defined in Expr. 1.

Lemma 10. *Let $\widetilde{G}^{\mathcal{M}_\lambda}$ be defined as in Algo. 5.4. For For any QPT adversary $\mathcal{M}_\lambda(\rho_\lambda)$ and any noticeable $\varepsilon(\lambda)$, it holds that*

$$\begin{aligned}&\left\{(\mathsf{OUT}^0,\mathsf{Val}^0) : (\mathsf{OUT}^0,\mathsf{Val}^0) \leftarrow \widetilde{G}^{\mathcal{M}_\lambda}(\lambda,\varepsilon,m_0,\rho_\lambda)\right\}\\ \overset{c}{\approx} &\left\{(\mathsf{OUT}^1,\mathsf{Val}^1) : (\mathsf{OUT}^1,\mathsf{Val}^1) \leftarrow \widetilde{G}^{\mathcal{M}_\lambda}(\lambda,\varepsilon,m_1,\rho_\lambda)\right\},\end{aligned} \tag{7}$$

where both ensembles are indexed by $\lambda \in \mathbb{N}$ and $(m_0,m_1) \in \{0,1\}^{\ell(\lambda)} \times \{0,1\}^{\ell(\lambda)}$.

Proof. We show this by means of a reduction to the VSS hiding game defined in Expr. 1. We assume for contradiction that there exist a machine $\mathcal{M}_\lambda(\rho_\lambda)$, a distinguisher $\mathcal{D}_\lambda$, and a pair of messages (m_0,m_1) so that Lem. 10 does not hold. We build a malicious $\mathcal{A}$ that wins the VSS hiding game (i.e., breaking Lem. 4).

The $\mathcal{A}$ works as follows:

1. It sends (m_0,m_1) to the external Ch for the VSS hiding game, as the Step 1 message of Expr. 1.
2. It internally samples a random size-k subset $\eta \subset [n]$.
3. It commits to η to the external Ch using ExtCom, as the Step 2 message of Expr. 1.
4. When the external Ch sends the Step 3 message of Expr. 1, it uses this message as the Step 1 message of the left session in its internal emulation of $\widetilde{G}^{\mathcal{M}_\lambda}$ with $\mathcal{M}_\lambda$.
5. It then decommits to η to the external Ch, as the Step 4 message of Expr. 1.
6. When it receives the revealed shares and their corresponding decommitment information form the external Ch (i.e., the Step 5 message of Expr. 1), it puts them together to define VI_η.
7. It executes the machine $\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)$ as in the **Remainder phase** of $\widetilde{G}^{\mathcal{M}_\lambda}$ (see Algo. 5.4).
8. It finally invokes the distinguisher $\mathcal{D}_\lambda$ on $\mathcal{SE}$'s output, and outputs whatever $\mathcal{D}_\lambda$ outputs.

Note that this $\mathcal{A}$ simulates perfectly emulates the execution of $\widetilde{G}^{\mathcal{M}_\lambda}(\lambda, \varepsilon, m_b, \rho_\lambda)$ (when the external Ch uses m_b). Note that in game $\widetilde{G}^{\mathcal{M}_\lambda}$, we define the variable VI_η by brute force (see Step 1b in Algo. 5.2). But in the above VSS hiding game, $\mathcal{A}$ learns the values in VI_η from the external Ch. This is only a syntax change as these two way leads to the same VI_η.

Therefore, if Lem. 10 does not hold, the above $\mathcal{A}$ will win the VSS hiding game with advantage non-negligibly greater than $1/2$.

□

Derivation of Contradiction. We assume for contradiction that Eq. (5) does not hold (i.e., the non-malleability of Prot. 3 does not hold). This means that there must be a (possibly non-uniform) QPT distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda, \sigma_\lambda\}_{\lambda\in\mathbb{N}}$, an ensemble of messages $\{(m_0, m_1)\}_{\lambda\in\mathbb{N}}$ and a function $\delta(\lambda) = 1/\mathsf{poly}(\lambda)$ such that for infinitely many $\lambda \in \mathbb{N}$, it holds that

$$\left|\Pr\left[\mathcal{D}_\lambda\left(\mathsf{OUT}^0, \mathsf{val}_{\widetilde{d}^0}(\widetilde{\tau}^0); \sigma_\lambda\right) = 1\right] - \Pr\left[\mathcal{D}_\lambda\left(\mathsf{OUT}^1, \mathsf{val}_{\widetilde{d}^1}(\widetilde{\tau}^1); \sigma_\lambda\right) = 1\right]\right| \geq \delta(\lambda), \tag{8}$$

where the first probability is taken over the random procedure $(\mathsf{OUT}^0, \widetilde{\tau}^0, \widetilde{d}^0) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m_0, \rho_\lambda)$, and the second probability is taken over the random procedure $(\mathsf{OUT}^1, \widetilde{\tau}^1, \widetilde{d}^1) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m_1, \rho_\lambda)$ (and the randomness due to the measurements performed by $\mathcal{D}_\lambda$).

Now recall that by Lem. 8, we have that for any m and any noticeable $\varepsilon_1(\lambda)$, it holds that

$$\begin{aligned}
&\left\{\left(\mathsf{OUT}^H, \mathsf{val}_{\widetilde{d}^H}(\widetilde{\tau}^H)\right) \;:\; (\mathsf{OUT}^H, \widetilde{\tau}^H, \widetilde{d}^H) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)\right\} \\
\overset{\mathrm{c}}{\approx}_{\varepsilon_1} &\left\{\left(\mathsf{OUT}^{\widetilde{H}}, \mathsf{val}_{\widetilde{d}^{\widetilde{H}}}(\widetilde{\tau}^{\widetilde{H}})\right) \;:\; (\mathsf{OUT}^{\widetilde{H}}, \widetilde{\tau}^{\widetilde{H}}, \widetilde{d}^{\widetilde{H}}) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}(\lambda, \varepsilon_1, m, \rho_\lambda\right\}.
\end{aligned}$$

Using this to replace terms on both sides of Inequality (8), we get

$$\left|\Pr\left[\mathcal{D}_\lambda\left(\mathsf{OUT}^0, \mathsf{val}_{\widetilde{d}^0}(\widetilde{\tau}^0); \sigma_\lambda\right) = 1\right] - \Pr\left[\mathcal{D}_\lambda\left(\mathsf{OUT}^1, \mathsf{val}_{\widetilde{d}^1}(\widetilde{\tau}^1); \sigma_\lambda\right) = 1\right]\right| \geq \delta(\lambda) - 2\varepsilon_1(\lambda), \tag{9}$$

where the inputs to $\mathcal{D}_\lambda$ in the above are sampled as $(\mathsf{OUT}^0, \widetilde{\tau}^0, \widetilde{d}^0) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}(\lambda, \varepsilon_1, m_0, \rho_\lambda)$ and $(\mathsf{OUT}^1, \widetilde{\tau}^1, \widetilde{d}^1) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}(\lambda, \varepsilon_1, m_1, \rho_\lambda)$.

Further, we have from Corollary 4 that for any m and any noticeable $\varepsilon_2(\lambda)$, there exists a noticeable $\varepsilon_1(\lambda) \leq 8\varepsilon_2(\lambda)$ that is efficiently computable from $\varepsilon_2(\lambda)$ such that

$$\begin{aligned}
&\left\{\left(\mathsf{OUT}^{\widetilde{H}}, \mathsf{val}_{\widetilde{d}^{\widetilde{H}}}(\widetilde{\tau}^{\widetilde{H}})\right) : (\mathsf{OUT}^{\widetilde{H}}, \widetilde{\tau}^{\widetilde{H}}, \widetilde{d}^{\widetilde{H}}) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}(\lambda, \varepsilon_1, m, \lambda, \rho_\lambda)\right\} \\
\overset{\mathrm{c}}{\approx}_{\varepsilon_2} &\left\{\left(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}\right) : (\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}} \leftarrow \widetilde{G}^{\mathcal{M}_\lambda}(\lambda, \varepsilon_2, m, \lambda, \rho_\lambda)\right\}.
\end{aligned}$$

Again, replacing terms on both sides of Inequality (9), we have

$$\left|\Pr\left[\mathcal{D}_\lambda\left(\mathsf{OUT}^0_{\mathcal{SE}}, \mathsf{Val}^0_{\mathcal{SE}}; \sigma_\lambda\right) = 1\right] - \Pr\left[\mathcal{D}_\lambda\left(\mathsf{OUT}^1_{\mathcal{SE}}, \mathsf{Val}^1_{\mathcal{SE}}; \sigma_\lambda\right) = 1\right]\right| \geq \delta(\lambda) - \varepsilon_1(\lambda) - \varepsilon_2(\lambda), \tag{10}$$

where the inputs to $\mathcal{D}_\lambda$ in the above are sampled as $(\mathsf{OUT}^0_{\mathcal{SE}}, \mathsf{Val}^0_{\mathcal{SE}}) \leftarrow \widetilde{G}^{\mathcal{M}_\lambda}(\lambda, \varepsilon_2, m_0, \lambda, \rho_\lambda)$ and $(\mathsf{OUT}^0_{\mathcal{SE}}, \mathsf{Val}^0_{\mathcal{SE}}) \leftarrow \widetilde{G}^{\mathcal{M}_\lambda}(\lambda, \varepsilon_2, m_1, \lambda, \rho_\lambda)$.

If we set $\varepsilon_2 := \frac{\delta}{18}$, then the lower-bound in Inequality (10) becomes

$$\delta(\lambda) - \varepsilon_1(\lambda) - \varepsilon_2(\lambda) \geq \delta(\lambda) - 8\varepsilon_2(\lambda) - \varepsilon_2(\lambda) = \delta(\lambda) - 9\varepsilon_2(\lambda) = \delta(\lambda) - 9 \cdot \frac{\delta(\lambda)}{18} = \frac{\delta(\lambda)}{2}.$$

Since $\frac{\delta(\lambda)}{2}$ is noticeable, this is in direct contradiction to Lem. 10, which says that the LHS of Inequality (10) can at most be negligible. We conclude that Inequality (8) is false.

This establishes that our protocol described in Prot. 3 is indeed non-malleable as per Def. 5 with $k = 1$.

6 Simulation-less Extractor $\mathcal{K}$: 1-1 Settings

In this section we introduce a 'basic' extractor machine $\mathcal{K}$. We then describe its operation and show that the stated properties hold.

We will rely on some of the notation from the previous section. In particular, recall from Algo. 5.2 that the procedure $\widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$ generates η, VI_η, and $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$. We will define $\mathsf{pref}' := (\mathsf{pref}, \eta, \mathsf{VI}_\eta)$. Also, we define the following quantity $p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1]$ that will be important in the statement of $\mathcal{K}$:

$$p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] := \Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right]. \tag{11}$$

Lemma 11 (Simulation-less Extraction). *Let $\widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$ be as defined in Algo. 5.2. There exists a QPT machine $\mathcal{K}$ such that for any noticeable $\varepsilon(\lambda)$, there is a noticeable $\varepsilon_1(\lambda) \leq \varepsilon(\lambda)$ that can be efficiently computed form ε, such that for any noticeable $\varepsilon_2(\lambda)$ and any tuple $\mathsf{pref}' = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau}, \eta, \mathsf{VI}_\eta)$ in the support of $\widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$, the following holds*

1. **(Almost Uniqueness:)** *$\mathcal{K}$ takes as input $(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$. It outputs a value $\mathsf{Val} \in \{0,1\}^{\ell(\lambda)} \cup \{\bot\}$ such that*

$$\Pr\left[\mathsf{Val} \notin \{\mathsf{val}(\widetilde{\tau}), \bot\} \ : \ \mathsf{Val} \to \mathcal{K}(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right] \leq \varepsilon_2(\lambda) + \mathsf{negl}(\lambda).$$

2. **(Extraction:)** *If $p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] \geq \varepsilon(\lambda)$, then it holds that*

$$\Pr\left[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right] \geq \frac{\varepsilon'(\lambda) - \varepsilon_2(\lambda)}{\widetilde{t}},$$

where $p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1]$ is defined in Eq. (11) and $\varepsilon'(\lambda) := \frac{\varepsilon(\lambda)}{10\widetilde{t}^2}$.

In the following, we fix a noticeable function $\varepsilon(\lambda)$ for which we want to prove Lem. 11. We show that it suffices to set $\varepsilon_1(\lambda) := \frac{\widetilde{t}+1}{\widetilde{t}^2+4\widetilde{t}+2} \cdot \varepsilon'(\lambda)$.

6.1 Description of $\mathcal{K}$

Before describing the extractor $\mathcal{K}$, we first need to introduce some new machines related to $\mathcal{G}_1$.

Algorithm 6.1: Machine $\mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$

Machine $\mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$: Recall that we have already defined $\mathcal{G}_1$ in Algo. 5.3. For $i \in [\widetilde{t}]\backslash\{1\}$, the machine $\mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$ works identically to $\mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$, apart from the following changes:

1. It commits to the value $i||\widetilde{x}_i$ instead of the value $1||\widetilde{x}_1$ in Step 2c of the right session.
   In more detail, $\mathcal{G}_i$ first prepares n views $\{\mathsf{rv}^{(2)}_j\}_{j\in[n]}$, corresponding to an MitH execution for the $(n+1, k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string $i||\widetilde{x}_i$, and then commits to each $\mathsf{rv}^{(2)}_j$ independently in parallel, using ExtCom.
2. Additionally, now the string $i||\widetilde{x}_i$ is used as the 'effective input' in the (virtual) MPC execution computing F^R_{consis} in Step 2d. (Indeed, this is an implicit change and occurs automatically when the first change is made.)

Next we define another machine $\mathcal{K}_i$ for each $i \in [\widetilde{t}]$ in Algo. 6.2. These $\mathcal{K}_i$'s sever as the basic component for the eventual extractor $\mathcal{K}$ we are going to build.

Algorithm 6.2: Machine $\mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$

Machine $\mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$**:** For each $i \in [\widetilde{t}]$, the machine $\mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$ works identically to machine $\mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$ except that

- In Step 3 of the right session, instead of following the honest receiver's algorithm, it invokes $\mathcal{SE}_{\mathsf{ExtCom}}(1^\lambda, 1^{\varepsilon_2^{-1}})$ to obtain an extracted value $\widetilde{v}$.
  In more detail, the shares $\{\mathsf{cv}_j^{(2)}\}_{j\in[n]}$ are committed by $\mathcal{M}_\lambda$ in Step 3 of the right session using independent ExtCom in parallel. $\mathcal{K}_i$ will extract all of these shares using the parallel extractability of ExtCom with ε_2-simulation (as per Def. 4), and compute $\widetilde{v} \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cv}_1^{(2)}, \ldots, \mathsf{cv}_n^{(2)})$.

Outputs of $\mathcal{K}_i$**:** To aid in our proof, we define the output of the machines $\mathcal{K}_i$ differently from the outputs of the machines described so far.
Let $\widetilde{v}$ denote the value extracted and recorded by $\mathcal{K}_i$ in Step 3. As described, $\mathcal{K}_i$ will complete the execution of both left and right sessions (just as in $\mathcal{G}_i$). Recall that we use $\widetilde{d}$ to denote the acceptance or rejection of the right-session honest receiver (i.e., its verdict). The output of $\mathcal{K}_i$ is denoted as $\mathsf{Val} \in \{0,1\}^{\ell(\lambda)} \cup \{\bot_{\widetilde{Y}}, \bot_{\mathsf{invalid}}\}$ (where $(\bot_{\widetilde{Y}}, \bot_{\mathsf{invalid}})$ are two specialized abort symbols), and is computed as follows:

1. If $\widetilde{d} = \top$. Then, there are two sub-cases:
   (a) $\widetilde{v} \notin \{\widetilde{x}_i\}_{i\in[\widetilde{t}]}$: In this case, we set $\mathsf{Val} \coloneqq \widetilde{v}$.
   (b) $\widetilde{v} \in \{\widetilde{x}_i\}_{i\in[\widetilde{t}]}$: In this case, we set $\mathsf{Val} \coloneqq \bot_{\widetilde{Y}}$.
2. Otherwise, if $\widetilde{d} = \bot$, set $\mathsf{Val} \coloneqq \bot_{\mathsf{invalid}}$.

We emphasize that such a Val satisfies the *syntactic* requirement in Property 1 of Lem. 11 [a] (but does not imply the actual probabilistic condition, which we will show separately).

[a] Note that here we defined two types of abortion: $\bot_{\widetilde{Y}}$ and $\bot_{\mathsf{invalid}}$, while Property 1 of Lem. 11 only allows a single abortion symbol $\bot$. We remark that this is only a cosmetic difference—It can be made consistent using the following rules: $\bot = \bot_{\widetilde{Y}}$ *and* $\bot = \bot_{\mathsf{invalid}}$ (i.e., $\mathsf{Val} = \bot \Leftrightarrow (\mathsf{Val} = \bot_{\widetilde{Y}} \vee \mathsf{Val} = \bot_{\mathsf{invalid}})$).

Finally, we are ready to define the extractor $\mathcal{K}$. Intuitively, $\mathcal{K}$ can be thought of as an average-case version of $\{\mathcal{K}_i\}_{i\in[\widetilde{t}]}$:

- **Extractor** $\mathcal{K}$**:** On input $(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$, $\mathcal{K}$ samples an index $i \xleftarrow{\$} [\widetilde{t}]$ uniformly and runs $\mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$, and outputs the resulting output of $\mathcal{K}_i$.

It is easy to see that the extractor $\mathcal{K}$ runs in polytime, and hence is a QPT machine. We now show the other properties in Lem. 11 are satisfied as well.

6.2 Almost Uniqueness of $\mathcal{K}$

In this part, we prove Property 1 of Lem. 11.

First, note that $\mathcal{K}$ by definition samples a random i and run $\mathcal{K}_i$. Thus, to prove Property 1, it suffices to prove the inequality shown in Property 1 for all $\mathcal{K}_i$'s. That is, to prove Property 1, it suffices to show the following: Under the same parameter settings as in Lem. 11, it holds that

$$\forall i \in [\widetilde{t}],\ \Pr\left[\mathsf{Val} \notin \{\mathsf{val}(\widetilde{\tau}), \bot\} \ :\ \mathsf{Val} \leftarrow \mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right] \leq \varepsilon_2(\lambda) + \mathsf{negl}(\lambda). \tag{12}$$

In the following, we focus on establishing Inequality (12).

First, note that the following holds for any $i \in [\widetilde{t}]$ (all the provabilities below is taken over the execution $\mathsf{Val} \leftarrow \mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$):

$$\begin{aligned}
\Pr[\mathsf{Val} \notin \{\mathsf{val}(\widetilde{\tau}), \bot\}] &= \Pr\left[\mathsf{Val} \notin \{\mathsf{val}(\widetilde{\tau}), \bot_{\mathsf{invalid}}, \bot_{\widetilde{Y}}\}\right] && (13)\\
&= \Pr\left[\left(\mathsf{Val} \notin \{\mathsf{val}(\widetilde{\tau}), \bot_{\mathsf{invalid}}, \bot_{\widetilde{Y}}\}\right) \wedge \left(\widetilde{d} = \top\right)\right] && (14)\\
&\leq \Pr\left[\left(\mathsf{Val} \notin \{\mathsf{val}(\widetilde{\tau}), \bot_{\widetilde{Y}}\}\right) \wedge \left(\widetilde{d} = \top\right)\right], && (15)
\end{aligned}$$

where Eq. (13) follows from the fact that the symbol $\bot$ corresponds to both $\bot_{\mathsf{invalid}}$ and $\bot_{\widetilde{Y}}$ (recall it from Algo. 6.2), Eq. (14) follows from the fact that Val is set to $\bot_{\mathsf{invalid}}$ on the right whenever $\widetilde{d} = \bot$ (see Algo. 6.2).

Thus, to prove Inequality (12), it suffices to upper-bound the RHS of Inequality (15) by $\varepsilon_2(\lambda) + \mathsf{negl}(\lambda)$. Towards that, we now compare the RHS of Inequality (15) with the corresponding condition on the *committed* value in Step 3 on the right in machine $\mathcal{G}_i$ (see Algo. 6.1). Recall that $\mathcal{K}_i$ differs from $\mathcal{G}_i$ only by its invocation of the $\mathcal{SE}_{\mathsf{ExtCom}}$ with error parameter ε_2 in the right Step 3. Thus, it follows from the extractability with ε_2-simulation of the right Step 3 that

$$\begin{aligned}
&\forall i \in [\widetilde{t}],\ \Pr\left[\left(\mathsf{Val} \notin \{\mathsf{val}(\widetilde{\tau}), \bot_{\widetilde{Y}}\}\right) \wedge \left(\widetilde{d} = \top\right) : \mathsf{Val} \leftarrow \mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right]\\
&\quad\leq \Pr\left[\left(\widetilde{\alpha} \notin \{\mathsf{val}(\widetilde{\tau})\} \cup \{\widetilde{x}_j\}_{j \in [\widetilde{t}]}\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] + \varepsilon_2, \qquad (16)
\end{aligned}$$

where $\widetilde{\alpha}$ denotes the value statistically bound (i.e., the committed value) in Step 3 of the right in machine $\mathcal{G}_i$.

Inequality (16) essentially reduces the almost uniqueness of $\mathcal{K}_i$ to that of machine $\mathcal{G}_i$. That is, we claim that to prove Inequality (12), it suffices to prove the following Lem. 12.

Lemma 12 (Almost Uniqueness of $\mathcal{G}_i$). *For $\mathcal{G}_i$ as defined, we have that*

$$\forall i \in [\widetilde{t}],\ \Pr\left[\left(\widetilde{\alpha} \notin \{\mathsf{val}(\widetilde{\tau})\} \cup \{\widetilde{x}_j\}_{j \in [\widetilde{t}]}\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \leq \mathsf{negl}(\lambda).$$

Proof of Lem. 12 (Sketch). This proof follows from standard techniques. Thus, we only provide a sketch.

At a high level, we prove this lemma by a reduction to the soundness of the commit-and-prove protocol shown in Prot. 1. Assuming Lem. 12 is false, we can build a malicious C^* that first commit to the value $\widetilde{\alpha}$ (by forwarding $\mathcal{M}$'s commitment in Step 3 of the right session to the external honest receiver), and then convince the external receiver that F^C_{consis} is satisfied with non-negligible probability. However, this should not happen because, by the condition in Lem. 12, $\widetilde{\alpha} \notin \{\mathsf{val}(\widetilde{\tau})\} \cup \{\widetilde{x}_j\}_{j \in [\widetilde{t}]}$ and thus F^C_{consis} is *not* satisfied.

The only caveat is: the external receiver will specify a random challenge set η, but in machine $\mathcal{G}_i$ this set η is determined by the trapdoor coin-flipping in Step 5b. To make sure that we can indeed employ the internal right session with $\mathcal{M}$ to convince the external receiver, we have to make sure that the internal right session uses the η sampled by the external receiver. To do that, first notice that if $\widetilde{\alpha} \notin \{\widetilde{x}_j\}_{j \in [\widetilde{t}]}$, then the trapdoor predicate $\phi(\cdot)$ in Step 5b in the right session will not be satisfied. By the security of the trapdoor coin-flipping protocol (in particular, Property 4 in Def. 24), there must exist a simulator $\mathcal{S}$ that can 'enforce' the coin-flipping result to a random η sampled independently. Using this $\mathcal{S}$, we can make sure that the internal $\mathcal{M}$ indeed generates a proof of consistency using the external receiver's η.

□

With Lem. 12 in hand, it is straightforward to see that Inequalities (15) and (16) and Lem. 12 together immediately implies Inequality (12).

This finishes the proof for Property 1 in Lem. 11.

6.3 Extraction Property of $\mathcal{K}$

In this part, we prove Property 2 of Lem. 11.

By definition, $\mathcal{K}$ simply picks an i uniformly from $[\widetilde{t}]$ and runs machine $\mathcal{K}_i$. Thus, to establish Property 2 in Lem. 11, it suffices to show the following:

Lemma 13. *For the same parameter settings as in Lem. 11, it holds that*

$$\exists i \in [\widetilde{t}],\ \Pr\left[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right] \geq \varepsilon'(\lambda) - \varepsilon_2(\lambda).$$

We claim that Lem. 13 follows as a result of the following Lem. 14 regarding machine $\mathcal{G}_i$'s defined in Algo. 6.1.

Lemma 14 (Validity of $\mathcal{G}_i$). *For the same parameter settings as in Lem. 11, it holds that*

$$\exists i \in [\widetilde{t}],\ \Pr\left[\left(\widetilde{\alpha} = \mathsf{val}(\widetilde{\tau})\right) \wedge \left(\widetilde{d} = \top\right)\ :\ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \geq \varepsilon'(\lambda).$$

In the following, we first prove Lem. 13, assuming Lem. 14 holds. Then, we will show the proof of Lem. 14 in Sec. 6.4, which represents the main technical task we perform in this section.

Proof. (Proving Lem. 13) We will show this via contradiction. For the sake of contradiction, assume that Lem. 13 does not hold. That is, we assume that under the parameter conditions in Lem. 13, it holds that

$$\forall i \in [\widetilde{t}],\ \Pr\left[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right] < \varepsilon'(\lambda) - \varepsilon_2(\lambda). \tag{17}$$

First, recall from the description of machine $\mathcal{K}_i$ (Algo. 6.2) that Val is set to $\perp_{\mathsf{invalid}}$ if $\widetilde{d} = \perp$. Thus, it must hold that

$$\begin{aligned} &\Pr\left[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right] \\ =\ &\Pr\left[\left(\widetilde{v} = \mathsf{val}(\widetilde{\tau})\right) \wedge \left(\widetilde{d} = \top\right) : \mathsf{Val} \leftarrow \mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right]. \end{aligned} \tag{18}$$

Next, recall that the only different between $\mathcal{K}_i$ and $\mathcal{G}_i$ lies in that $\mathcal{K})_i$ additionally invoke the extractor with error parameter ε_2 in Step 3 of the right session. By the ε_2-simulatable extractability of ExtCom (as per Def. 4), it holds hat

$$\begin{aligned} &\Big|\Pr\left[\left(\widetilde{\alpha} = \mathsf{val}(\widetilde{\tau})\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \\ &\quad - \Pr\left[\left(\mathsf{Val} = \mathsf{val}(\widetilde{\tau})\right) \wedge \left(\widetilde{d} = \top\right) : \mathsf{Val} \leftarrow \mathcal{K}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right]\Big| \leq \varepsilon_2(\lambda) \end{aligned} \tag{19}$$

Inequality (19) Eq. (18), and Inequality (17) together imply the following

$$\forall i \in [\widetilde{t}],\ \Pr\left[\left(\widetilde{\alpha} = \mathsf{val}(\widetilde{\tau})\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(\mathsf{pref}')\right] < \varepsilon'(\varepsilon),$$

which contradicts Lem. 14.

This completes the proof of Lem. 13. □

This completes the proof for Property 2 of Lem. 11, modulo the proof of Lem. 14 that we will present in Sec. 6.4.

6.4 Validity of $\mathcal{G}_i$

In this part, we present the proof for Lem. 14.

We first need to define (in Algo. 6.3) two helper machines $\mathcal{G}'_i$ and $\mathcal{G}''_i$ ($\forall i \in [\widetilde{t}]$). They are machines very similar to the $\mathcal{G}_i$.

Algorithm 6.3: Machines $\mathcal{G}'_i$ and $\mathcal{G}''_i$

Machine $\mathcal{G}'_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$: For each $i \in [\widetilde{t}]$, machine $\mathcal{G}'_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$ proceeds as follows:

1. It behaves identically as $\mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$ (see Algo. 6.1) until the end of Step 2b.
2. It then performs *brute-force computation* to obtain the VSS shares $\{\mathsf{rv}_i^{(1)}\}_{i\in[\widetilde{t}]}$ committed by $\mathcal{M}_\lambda$ in the Step 2b Naor's commitment of the left session, and then runs the reconstruction algorithm $\mathsf{VSS}_{\mathsf{Recon}}$ to obtain the puzzle solutions $x_1||\ldots||x_t$ (it aborts if reconstruction is unsuccessful).
3. It then samples an uniform index $s \xleftarrow{\$} [t]$ and commits to the value $(s||x_s)$ (i.e., it uses the s-th puzzle solution obtained from the previous step) in the left Step 3.
   In more detail, it prepares n views $\{\mathsf{cv}_i^{(2)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1, 2k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the message $(s||x_s)$. It commits to each $\mathsf{cv}_i^{(2)}$ ($i \in [n]$) independently in parallel, using ExtCom, as the Step 3 message of the left session.
4. In Step 4 of the left session, $\mathcal{G}'_i$ checks the extracted x_s is indeed the s-th puzzle solution as revealed by $\mathcal{M}_\lambda$. If not, it aborts.
5. In Step 5a of the left session, it uses $(s\|x_s)$ as the 'effective input' in the virtual execution of F^C_{consis}. This is possible because x_s is indeed the s-th puzzle solution, and thus $s\|x_s$ serves as a valid witness for the 'second clause' of F^C_{consis}.
6. All other steps are carried out as in $\mathcal{G}_i$.

Machine $\mathcal{G}''_i(1^\lambda, \mathsf{pref}')$: For each $i \in [\widetilde{t}]$, $\mathcal{G}''_i(1^\lambda, \mathsf{pref}')$ works similarly to $\mathcal{G}'_i$ except that $\mathcal{G}''_i$ no longer runs the extractor $\mathcal{SE}_{\mathsf{ExtCom}}(1^\lambda, 1^{\varepsilon_1^{-1}})$ in the left Step 2c, and thus does not need the error parameter ε_1 anymore.

At a High Level. we prove Lem. 14 by contradiction. Assuming Lem. 14 is false, we will derive the desired contradiction using the machine $\mathcal{G}''_1$. In particular, we will prove an upper-bound and a lower-bound for the probability related to the committed value in the right Step 3 in $\mathcal{G}''_1$. We will show that these two bounds indeed contradict to each other, thus finishing the proof of Lem. 14.

In the following, we first present the upper-bound in Lem. 15 and the lower-bound in Lem. 16 without a proof, and show how to derive the desired contradiction if these bounds hold. We then focus on establishing these bounds in Sec. 6.5 and 6.6 respectively.

The Bounds. The upper-bound is for the probability of the event that the value $\widetilde{\alpha}$ committed by $\mathcal{M}_\lambda$ in the right Step 3 in $\mathcal{G}''_1$ is 'valid', i.e., it is either the message committed to initially on the right (namely, $\mathsf{val}(\widetilde{\tau})$), or a legitimate puzzle solution (i.e., is among the strings $(\widetilde{x}_1, \ldots, \widetilde{x}_{\widetilde{t}})$). We capture this formally in the following Lem. 15, and prove it in Sec. 6.5. We remark that Lem. 15 does not rely on the assumption (for contradiction) that Lem. 14 is false.

Lemma 15 (Upper Bound). *For the same parameter settings as in Lem. 14, it holds that*

$$\Pr\left[\left(\widetilde{\alpha} \in \{\mathsf{val}(\widetilde{\tau})\} \cup \{\widetilde{x}_j\}_{j\in[\widetilde{t}]}\right) \wedge \left(\widetilde{d} = \top\right) \ : \ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\right] \leq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] + \varepsilon_1 + \mathsf{negl}(\lambda).$$

The lower bound is for the probability of the event that the value $\widetilde{\alpha}$ committed by $\mathcal{M}_\lambda$ in the right Step 3 in $\mathcal{G}_1''$ is actually a puzzle solution (i.e., is among the strings $(\widetilde{x}_1, \ldots, \widetilde{x}_{\widetilde{t}})$). This is formally stated as the following Lem. 16. We present its proof in Sec. 6.6. We remark that Lem. 16 relies on the assumption (for contradiction) that Lem. 14 is false.

Lemma 16 (Lower bound). *Assume that Lem. 14 is false. Then, for the same parameter settings as in Lem. 14, it holds that*

$$\forall i \in [\widetilde{t}], \ \Pr\left[\left(\widetilde{\alpha} = \widetilde{x}_i\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\right] \geq \frac{p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1 - \varepsilon'}{t} - \varepsilon_1 - \mathsf{negl}(\lambda).$$

The Final Contradiction. Assume Lem. 14 is false. Using Lem. 15 and 16, we derive the desired contradiction in the following. All the probabilities below are taken over $(\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')$, which we omit for notation succinctness.

$$\begin{aligned}
\Pr\left[\left(\widetilde{\alpha} \in \{\mathsf{val}(\widetilde{\tau})\} \cup \{\widetilde{x}_j\}_{j\in[\widetilde{t}]}\right) \wedge \left(\widetilde{d} = \top\right)\right] &= \Pr\left[\left(\widetilde{\alpha} = \mathsf{val}(\widetilde{\tau})\right) \wedge \left(\widetilde{d} = \top\right)\right] + \sum_{i=1}^{\widetilde{t}} \Pr\left[\left(\widetilde{\alpha} = x_i\right) \wedge \left(\widetilde{d} = \top\right)\right] \\
&\geq \sum_{i=1}^{\widetilde{t}} \Pr\left[\left(\widetilde{\alpha} = x_i\right) \wedge \left(\widetilde{d} = \top\right)\right] \\
&\geq \widetilde{t} \cdot \left(\frac{p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1 - \varepsilon'}{t} - \varepsilon_1 - \mathsf{negl}(\lambda)\right) \qquad (20) \\
&= \widetilde{t} \cdot \frac{1}{t} \cdot \left(p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - (t+2)\varepsilon_1 - \varepsilon'\right) - \mathsf{negl}(\lambda) \\
&\geq \frac{t+1}{t} \cdot \left(p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - (t+2)\varepsilon_1 - \varepsilon'\right) - \mathsf{negl}(\lambda) \qquad (21) \\
&= \frac{t+1}{t} \cdot \left(p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - (t+2)\varepsilon_1 - \varepsilon' - \frac{t}{t+1} \cdot \varepsilon_1\right) + \varepsilon_1 - \mathsf{negl}(\lambda) \\
&= \frac{t+1}{t} \cdot \left(p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - \frac{t^2+4t+2}{t+1} \cdot \varepsilon_1 - \varepsilon'\right) + \varepsilon_1 - \mathsf{negl}(\lambda) \\
&= \frac{t+1}{t} \cdot \left(p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon'\right) + \varepsilon_1 - \mathsf{negl}(\lambda) \qquad (22) \\
&= p^{\mathsf{Sim}}_{\mathsf{pref}}[\varepsilon_1] + \varepsilon_1 + \left(\frac{p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1]}{t} - 2\varepsilon' - \frac{2\varepsilon'}{t}\right) - \mathsf{negl}(\lambda) \\
&\geq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] + \varepsilon_1 + \frac{5t^2 - t - 1}{5t^3} \cdot \varepsilon - \mathsf{negl}(\lambda), \qquad (23)
\end{aligned}$$

where Inequality (20) follows from Lem. 16, Inequality (21) follows from the assumption that $\widetilde{t} \geq t+1$, Eq. (22) follows from our parameter setting $\varepsilon_1(\lambda) = \frac{t+1}{t^2+4t+2} \cdot \varepsilon'(\lambda)$, and Inequality (23) follows from the assumption that $p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] \geq \varepsilon(\lambda)$ and our parameter setting $\varepsilon'(\lambda) = \frac{\varepsilon(\lambda)}{10t^2}$.

Recall that t is the tag taking values from $[n]$ with n being a polynomial of λ. Also recall that $\varepsilon(\lambda)$ is an inverse polynomial on λ. Therefore, Inequality (23) can be written as:

$$\Pr\left[\left(\widetilde{\alpha} \in \{\mathsf{val}(\widetilde{\tau})\} \cup \{\widetilde{x}_j\}_{j\in[\widetilde{t}]}\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\right] \geq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] + \varepsilon_1(\lambda) + \frac{1}{\mathsf{poly}(\lambda)} - \mathsf{negl}(\lambda),$$

which contradicts the upper-bound shown in Lem. 15, yielding the desired contradiction.

This concludes the proof of Lem. 14.

6.5 The Upper Bound

In this part, we present the proof for Lem. 15.

We start by recalling from Eq. (11) that by definition:

$$\Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] = p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1]. \tag{24}$$

We will use the machines $\mathcal{G}_1'$ and $\mathcal{G}_1''$ (see Algo. 6.3). First, recall that compared with $\mathcal{G}_1$, $\mathcal{G}_1'$ performs a brute-force computation to learn $s\|x_s$ (as per Step 2); It commits to $s\|x_s$ in Step 3 of the left session (as per Step 3) and use $s\|x_s$ as the witness to perform the proof of consistency in Step 5 of the left session (as per Step 5). In other words, what $\mathcal{G}_1'$ does is simply to change the witness committed in Step 3 in the commit-and-prove protocol consisting of Step 3 and Step 5; Note that that 'witness' used by $\mathcal{G}_1'$ (i.e., $s\|x_s$) still satisfies the target predicate F^C_{consis}. This will not be noticed by $\mathcal{M}_\lambda$ as the commit-and-prove protocol is witness-indistinguishable[30]. Also note that the brute-force computation performed by $\mathcal{G}_1'$ does not affect the computational indistinguishability between $\mathcal{G}_1'$ and $\mathcal{G}_1$, as that step happens before the beginning of Step 3 and can be treated as non-uniform advice when invoking the witness-indistinguishability of the commit-and-prove protocol. This argument implies the following:

$$\left|\Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] - \Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1'(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right]\right| \leq \mathsf{negl}(\lambda). \tag{25}$$

Eq. (24) and Inequality (25) together imply the following:

$$\Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1'(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \leq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] + \mathsf{negl}(\lambda) \tag{26}$$

Next, notice that the difference between $\mathcal{G}_1'$ and $\mathcal{G}_1''$ is that the latter stops running machine $\mathcal{SE}_{\mathsf{ExtCom}}(1^\lambda, 1^{\varepsilon_1^{-1}})$ (see Algo. 6.3). Thus, $\mathcal{G}_1'$ and $\mathcal{G}_1''$ are at most ε_1-far. Therefore, Inequality (26) implies the following:

$$\Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\right] \leq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] + \varepsilon_1(\lambda) + \mathsf{negl}(\lambda) \tag{27}$$

Inequality (27) immediately implies the inequality in Lem. 15.

This completes the proof of Lem. 15.

[30] This can be proven formally using standard techniques and thus we omit the details. A formal proof can be found in, e.g., [CCLY22a, Section 6.5].

6.6 The Lower Bound

In this part, we present the proof for Lem. 16.

Assumption for Contradiction. As mentioned earlier, this proof (in particular, in the proof of Claim 8) will make use of the negation of Lem. 14, which is for the sake of contradiction. That is, we assume for contradiction that: Under the parameter setting of Lem. 14, it holds that

$$\forall i \in [\widetilde{t}],\ \Pr\Big[\big(\widetilde{\alpha} = \mathsf{val}(\widetilde{\tau})\big) \wedge \big(\widetilde{d} = \top\big)\ :\ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big] < \varepsilon'(\lambda) \tag{28}$$

Preparatory Claims. Before showing this bound, we will first consider three preparatory claims (i.e., Claims 7 to 9) regarding the machine $\mathcal{G}_i$'s. They will help us to bound the probability regarding $\mathcal{G}_1''$ as in Lem. 16 eventually.

Claim 7. *For the same parameter settings as in Lem. 16, it holds that*

$$\forall i \in [\widetilde{t}],\ \Pr\Big[\widetilde{d} = \top\ :\ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1(\lambda) - \mathsf{negl}(\lambda).$$

Proof. This lemma again makes use of the machines $\mathcal{G}_1'$ and $\mathcal{G}_1''$ defined in Algo. 6.3.

We first claim that

$$\forall i \in [\widetilde{t}],\ \Big|\Pr\Big[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big] - \Pr\Big[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i'(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big]\Big| \leq \mathsf{negl}(\lambda). \tag{29}$$

Indeed, we have already shown Inequality (29) for the case $i = 1$ (see Inequality (25)). Inequality (29) follows from the same argument as we presented for Inequality (25). Thus, we omit the details.

Similarly, we also have the following 'all-i' version of Inequality (27):

$$\forall i \in [\widetilde{t}],\ \Big|\Pr\Big[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i'(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big] - \Pr\Big[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i''(1^\lambda, \mathsf{pref}')\Big]\Big| \leq \varepsilon_1(\lambda) + \mathsf{negl}(\lambda). \tag{30}$$

Next, using a similar (non-uniform) argument as for Inequality (25) over the commit-and-prove protocol consisting of the hard puzzle setup step (i.e., Step 2) of the right session, we can establish the following

$$\forall i \in [\widetilde{t}],\ \Big|\Pr\Big[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i''(1^\lambda, \mathsf{pref}')\Big] - \Pr\Big[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\Big]\Big| \leq \mathsf{negl}(\lambda). \tag{31}$$

Using Inequalities (29) to (31) by setting $i = 1$, we obtain

$$\Big|\Pr\Big[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\Big]\Big| \geq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - \varepsilon_1(\lambda) - \mathsf{negl}(\lambda), \tag{32}$$

where recall the definition of $p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1]$ from Eq. (11).

Finally, we have the following for all $i \in [\widetilde{t}]$:

$$\begin{aligned}
&\Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \\
\geq\ &\Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}'_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] - \mathsf{negl}(\lambda) && (33)\\
\geq\ &\Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}''_i(1^\lambda, \mathsf{pref}')\right] - \varepsilon_1(\lambda) - \mathsf{negl}(\lambda) && (34)\\
\geq\ &\Pr\left[\widetilde{d} = \top : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}''_1(1^\lambda, \mathsf{pref}')\right] - \varepsilon_1(\lambda) - \mathsf{negl}(\lambda) && (35)\\
\geq\ &p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1(\lambda) - \mathsf{negl}(\lambda), && (36)
\end{aligned}$$

where Inequality (33) follows from Inequality (29), Inequality (34) follows from Inequality (30), Inequality (35) follows from Inequality (31), and Inequality (36) follows from Inequality (32).

This concludes the proof of Claim 7.

□

Claim 8. *Assume that Lem. 14 is false. For the same parameter settings as in Lem. 16, it holds that*

$$\forall i \in [\widetilde{t}],\ \Pr\left[\left(\widetilde{\alpha} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]}\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \geq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1(\lambda) - \varepsilon'(\lambda) - \mathsf{negl}(\lambda).$$

Proof. In the following, we will fix an arbitrary $i \in [\widetilde{t}]$ and describe events only within $\mathcal{G}_i$, and omit the rider that the variables in question are generated from $\mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$ for notation convenience.

First, it follows from Lem. 12 and Claim 7 that

$$\Pr\left[\left(\widetilde{\alpha} \in \{\mathsf{val}(\widetilde{\tau})\} \cup \{\widetilde{x}_j\}_{j\in[\widetilde{t}]}\right) \wedge \left(\widetilde{d} = \top\right)\right] \geq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1(\lambda) - \mathsf{negl}(\lambda). \tag{37}$$

Inequality (37) and Inequality (28) (i.e., our assumption for contradiction) together immediately imply

$$\Pr\left[\left(\widetilde{\alpha} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]}\right) \wedge \left(\widetilde{d} = \top\right)\right] \geq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1(\lambda) - \varepsilon'(\lambda) - \mathsf{negl}(\lambda). \tag{38}$$

This completes the proof of Claim 8.

□

Claim 9. *For the same parameter settings as in Lem. 16, it holds that*

$$\forall i \in [\widetilde{t}],\ \Pr\left[\left(\widetilde{\alpha} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]\setminus\{i\}}\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \leq \mathsf{negl}(\lambda),$$

Proof. We first define a new machine $\mathcal{G}^*_i$ as follows.

Algorithm 6.4: Machine $\mathcal{G}^*_i(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon^{*-1}}, \mathsf{pref}')$

This machine takes an additional parameter ε^* than $\mathcal{G}_i$. On input $(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon^{*-1}}, \mathsf{pref}')$, $\mathcal{G}^*_i$ is identical to $\mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$ except for the following difference:

- In Step 2a of the right session, it invokes the the extractor for ExtCom with simulation error set to ε^*, to extracts the set $\widetilde{\eta}$ committed by $\mathcal{M}$.
- In Step 2d of the right session, it prepares the shares $\{\widetilde{\mathsf{rv}}^{(3)}_j\}_{j\in[n]}$ differently. First, recall that each $\widetilde{\mathsf{rv}}^{(3)}_j$ is the output of the MitH party P_j using input $\widetilde{\mathsf{rv}}^{(1)}_j \| \widetilde{\mathsf{rv}}^{(2)}_j$ to compute the ideal functionality F^R_{consis}. $\mathcal{G}^*_i$ modifies the input to P_j's as follows:

- It defines a new vector $(\widetilde{x}_1^*, \ldots, \widetilde{x}_{\widetilde{t}}^*)$, where $\widetilde{x}_i^*$ is equal to the $\widetilde{x}_i$ sampled in Step 2b, but $\widetilde{x}_j^* = 0^\lambda$ for all $j \neq i$.
- It creates a new set of VSS shares $\{\widetilde{\mathsf{rv}}_j^{*(1)}\}_{j\in[n]}$ for the new $(\widetilde{x}_1^*, \ldots, \widetilde{x}_{\widetilde{t}}^*)$ satisfying the requirement that $\widetilde{\mathsf{rv}}_j^{*(1)} = \widetilde{\mathsf{rv}}_j^{(1)}$ for all $j \in \widetilde{\eta}$. That is, it takes the shares $\{\widetilde{\mathsf{rv}}_j^{(1)}\}_{j\in\widetilde{\eta}}$, which is the VSS shares for the original vector $(\widetilde{x}_1, \ldots, \widetilde{x}_{\widetilde{t}})$, and computes a new set of 'the remainder' shares $\{\widetilde{\mathsf{rv}}_j^{*(1)}\}_{j\in[n]\setminus\widetilde{\eta}}$, such that the new set $\{\widetilde{\mathsf{rv}}^{*(1)}\}_{i\in[n]\setminus\widetilde{\eta}} \cup \{\widetilde{\mathsf{rv}}^{(1)}\}_{i\in\widetilde{\eta}}$ constitutes VSS sharing of the new vector $(\widetilde{x}_1^*, \ldots, \widetilde{x}_{\widetilde{t}}^*)$. We remark that this is possible because any k shares of a $(n+1, k)$-VSS scheme contain no information of the underlying secret; Thus, any k shares of some secret can be 'extended' to n shares that constitute a VSS of a different secret. Indeed, the VSS scheme we use (from [CDD+99]) satisfies this property.

With the above, $\mathcal{G}_i^*$ prepares $\{\widetilde{\mathsf{rv}}_j^{(3)}\}_{j\in[n]}$ by running the (n, k)-MitH execution with party P_j $(\forall j \in [n])$ using $\widetilde{\mathsf{rv}}_j^{*(1)} \| \widetilde{\mathsf{rv}}_j^{(2)}$ as its input.

– It finishes the remaining execute in the same manner as $\mathcal{G}_i$.

We first claim that for any noticeable $\varepsilon^*(\lambda)$ and any $i \in [\widetilde{t}]$, it holds that

$$\begin{aligned} &\Pr\left[\widetilde{\alpha} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]\setminus\{i\}} \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \\ \overset{\mathrm{c}}{\approx}_{\varepsilon^*} &\Pr\left[\widetilde{\alpha} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]\setminus\{i\}} \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i^*(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon^{*-1}}, \mathsf{pref}')\right]. \end{aligned} \tag{39}$$

To see Eq. (39), note that the only difference between $\mathcal{G}_i$ and $\mathcal{G}_i^*$ is how the shares $\mathsf{rv}_j^{(3)}$'s are generated (modulo the extractor with ε^*-simulation that $\mathcal{G}_i^*$ invokes in Step 2a, which is already taken into account by the symbol $\overset{\mathrm{c}}{\approx}_{\varepsilon^*}$). First, we remark that the new $\mathsf{rv}_j^{*(1)}$'s generated by $\mathcal{G}_i^*$ still satisfy the predicate F_{consis}^R in Step 2d. This is because these $\mathsf{rv}_j^{*(1)}$'s will reconstruct to $(\widetilde{x}_1^*, \ldots, \widetilde{x}_{\widetilde{t}}^*)$, where $\widetilde{x}_i^*$ does equal to $\widetilde{x}_i$, to which the (unchanged) $\mathsf{rv}_j^{(2)}$'s will reconstruct. Thus, if we compare the $\{\mathsf{rv}_j^{(3)}\}_{j\in\widetilde{\eta}}$ shares between $\mathcal{G}_i$ and $\mathcal{G}_i^*$, they are the views of parties P_j $(j \in \widetilde{\eta})$ resulted from different inputs that lead to the same output for F_{consis}^R. By the (n, k)-privacy of the underlying MPC, we know that any k shares (i.e., those in set $\widetilde{\eta}$) does not reveal the input of other parties P_j for $j \in [n] \setminus \widetilde{\eta}$. Therefore, the view of $\mathcal{M}_\lambda$ is computationally indistinguishable between $\mathcal{G}_i$ and $\mathcal{G}_i^*$ (modulo the error ε^* accounting for the extractor with ε^*-simulation invoked by $\mathcal{G}_1^*$ in Step 2a).

This seems to already establish Eq. (39). But we remark that there is a caveat: what we have shown so far is about $\mathcal{M}_\lambda$ view. But the event in Eq. (39) is about the committed value $\widetilde{\alpha}$. To compensate for that, note that the committed $\widetilde{\alpha}$ can be efficiently extracted using the extractor $\mathcal{SE}$ for the ExtCom in Step 3 of the right session, with an arbitrarily small noticeable simulation error ε. Thus, the indistinguishability of $\mathcal{M}_\lambda$ does translate to the event regarding the committed $\widetilde{\alpha}$. This finises the proof of Eq. (39).

With Eq. (39) in hand (where note that the ε^* can be made arbitrarily small), to prove Claim 9, it suffices to prove the following regarding machine $\mathcal{G}^*$: for all $i \in [\widetilde{t}]$ and all ε^*, it holds that

$$\Pr\left[\widetilde{\alpha} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]\setminus\{i\}} \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i^*(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon^{*-1}}, \mathsf{pref}')\right] \leq \mathsf{negl}(\lambda). \tag{40}$$

In the following, we establish Inequality (40) by reducing it to the VSS hiding game shown in Expr. 1.

Assume for the sake of contradiction that Inequality (40) does not hold. Namely, there exists an $i \in [\widetilde{t}]$ and a inverse polynomial quantity $\nu(\lambda)$ such that for infinitely many $\lambda \in \mathbb{N}$, it holds that

$$\Pr\left[\widetilde{\alpha} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]\setminus\{i\}} \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i^*(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon^{*-1}}, \mathsf{pref}')\right] > \nu(\lambda).$$

We show there exists an adversary $\mathcal{A}$ that wins the VSS hiding game shown in Expr. 1. $\mathcal{A}$ works as follows:

1. $\mathcal{A}$ picks two tuples of messages $\{\widetilde{x}_j\}_{j\in[\widetilde{t}]}$ and $\{\widetilde{x}'_j\}_{j\in[\widetilde{t}]}$ that are distinct in every entry other than i (i.e., $\widetilde{x}_j \neq \widetilde{x}'_j \ \forall j \in [\widetilde{t}] \setminus \{i\}$), but it holds that $\widetilde{x}_i = \widetilde{x}'_i$. Externally, it sends $m_0 \coloneqq \{\widetilde{x}_j\}_{j\in[\widetilde{t}]}$ and $m_1 \coloneqq \{\widetilde{x}'_j\}_{j\in[\widetilde{t}]}$ to the challenger Ch for the VSS hiding game, as Step 1 in Expr. 1.
2. Internally $\mathcal{A}$ starts executing $\mathcal{G}_i^*(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon^{*-1}}, \mathsf{pref}')$. It proceeds with this execution till the end of Step 2a of the right session. Note that by definition of $\mathcal{G}_i^*$ (see Algo. 6.4), the set $\widetilde{\eta}$ has already been extracted at this moment. $\mathcal{A}$ then commit to $\widetilde{\eta}$ using ExtCom to the external challenger Ch as Step 2 in Expr. 1.
3. The challenger next sends commitments as Step 3 in Expr. 1. $\mathcal{A}$ forwards these commitments to $\mathcal{M}_\lambda$ (at Step 2b on the right).
4. $\mathcal{A}$ then sends the $\widetilde{\eta}$ together with the decommitment information w.r.t. its ExtCom made in Step 2, as the Step 4 message in Expr. 1.
5. Then, Ch will send the shares $\{\mathsf{v}_j\}_{j\in\widetilde{\eta}}$ to $\mathcal{A}$, as the Step 5 message in Expr. 1. $\mathcal{A}$ records this values.
6. Internally, $\mathcal{A}$ continues the execute as $\mathcal{G}_i^*$, until the beginning of Step 2d. It executes Step 2d in the following manner (which is also identical to $\mathcal{G}_i^*$ by renaming some variables as explained below):
   - It first prepares the shares $\{\mathsf{rv}_j^{*(1)}\}_{j\in[n]}$ that constitute a VSS of $(\widetilde{x}_1^*, \ldots, \widetilde{x}_{\widetilde{t}}^*)$ where $\widetilde{x}_i^* = \widetilde{x}_i$ but $\widetilde{x}_j^* = 0^\lambda$ for all $j \neq i$, and these shares satisfy the requirement that $\mathsf{rv}_j^{*(1)} = \mathsf{v}_j$ for $j \in \widetilde{\eta}$, as we explain in the description of $\mathcal{G}_i^*$ with $\{\mathsf{v}_j\}_{j\in\widetilde{\eta}}$ playing the role of $\{\mathsf{rv}_j^{(1)}\}_{j\in\widetilde{\eta}}$ in the description in Algo. 6.4.
   - $\mathcal{G}_i^*$ then prepares $\{\widetilde{\mathsf{rv}}_j^{(3)}\}_{j\in[n]}$ by running the (n,k)-MitH execution with party P_j $(\forall j \in [n])$ using $\widetilde{\mathsf{rv}}_j^{*(1)} \| \widetilde{\mathsf{rv}}_j^{(2)}$ as its input.

   It is worth noting that in this step, $\mathcal{A}$ does not make use of the shares $\{\mathsf{v}_j\}_{j\in[n]\setminus\widetilde{\eta}}$ (which are anyway only know to the external Ch but not to $\mathcal{A}$). $\mathcal{A}$ only uses the shares $\{\mathsf{v}_j\}_{j\in\widetilde{\eta}}$ that is revealed by Ch in Step 5.
7. $\mathcal{A}$ then internally finish the remaining execution in the same manner as $\mathcal{G}_i^*$, with only one difference—In Step 3 of the right session, $\mathcal{A}$ invokes the extractor $\mathcal{SE}_{\mathsf{ExtCom}}$ with error parameter $\varepsilon_{\mathcal{A}}$ to get an extracted value $\widetilde{v}$. It parses $\widetilde{v}$ as $j||a$. If $a = \widetilde{x}_j$, $\mathcal{A}$ halts and output $b' = 0$; If $a = \widetilde{x}'_j$, $\mathcal{A}$ halts and output $b' = 1$; If neither of the cases happen, $\mathcal{A}$ halts and output a random bit b'

By the above description, it is not hard to see that up to Step 6, the internal execution of $\mathcal{A}$ perfectly emulates the view of $\mathcal{M}_\lambda$ in game $\mathcal{G}_i^*$. If $\mathcal{M}_\lambda$ indeed commits to an $\widetilde{\alpha} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]\setminus\{i\}}$ (and $\widetilde{d} = \top$) with probability $\nu(\lambda)$, $\mathcal{A}$ in Step 7 will extract a $\widetilde{v} \in \{\widetilde{x}_j\}_{j\in[\widetilde{t}]\setminus\{i\}}$ (or $\widetilde{v} \in \{\widetilde{x}'_j\}_{j\in[\widetilde{t}]\setminus\{i\}}$, depending on the Ch uses m_0 or m_1) with probability at least $\nu(\lambda) - \varepsilon_{\mathcal{A}}(\lambda)$. Therefore, the advantage of $\mathcal{A}$ in the VSS hiding game is at least:

$$\big(\nu(\lambda) - \varepsilon_{\mathcal{A}}(\lambda)\big) \cdot 1 + \big(1 - (\nu(\lambda) - \varepsilon_{\mathcal{A}}(\lambda))\big) \cdot \frac{1}{2} = \frac{1}{2} + \frac{\nu(\lambda) - \varepsilon_{\mathcal{A}}(\lambda)}{2}.$$

Note that we can set $\varepsilon_{\mathcal{A}}(\lambda)$ to be an arbitrarily small noticeable function. By setting $\varepsilon_{\mathcal{A}}(\lambda) := \frac{\nu(\lambda)}{2}$, the above lower bound becomes $\frac{1}{2} + \frac{\nu(\lambda)}{4}$. Since $\frac{\nu(\lambda)}{4}$ is still a noticeable function, this contradicts Lem. 4, breaking the VSS hiding game.

This concludes the proof of Claim 9.

□

Finishing the Proof of Lem. 16. With the above preparatory Claims 7 to 9, we now proceed to finish the proof of Lem. 16.

From $\mathcal{G}_i$ to $\mathcal{G}'_i$. We start by comparing machine $\mathcal{G}_i$ and $\mathcal{G}'_i$ (see Algo. 6.3). Note that $\mathcal{G}'_i$ differs from $\mathcal{G}_i$ in that it performs brute-force computation to extract the puzzle solutions from Step 2b of the left session (see Step 2 of Algo. 6.3), while $\mathcal{G}_i$ extract a puzzle solution $(j\|x_j)$ using the extractability from ExtCom. Note that the $(j\|x_j)$ extracted by $\mathcal{G}_i$ must be among the t real solutions (which are all extracted by $\mathcal{G}'_i$ using brute force). Also, recall form Algo. 6.3 that $\mathcal{G}'_i$ picks a random $(s\|x_s)$ to finish the reminder execution as in $\mathcal{G}_i$. Thus, in the case where $j = s$ (i.e., $\mathcal{G}'_i$ happens to guess the same $(j\|x_j)$ as extracted by $\mathcal{G}_i$), then the games $\mathcal{G}_i$ and $\mathcal{G}'_i$ are *identical*. Moreover, since $\mathcal{G}'_i$ guesses s uniformly at random from $[t]$, the event $j = s$ happens with probability at least $1/t$. Therefore, the following holds:

$$\begin{aligned}\forall i \in [\widetilde{t}],\ \Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}'_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big] \\ \geq \frac{\Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big]}{t}.\end{aligned} \tag{41}$$

On the other hand, notice that Claim 8 and Claim 9, we conclude that

$$\forall i \in [\widetilde{t}],\ \Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge \big(\widetilde{d} = \top\big) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1(\lambda) - \varepsilon'(\lambda) - \mathsf{negl}(\lambda). \tag{42}$$

Inequalities (42) and (41) together imply the following:

$$\forall i \in [\widetilde{t}],\ \Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}'_i(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\Big] \geq \frac{p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1(\lambda) - \varepsilon'(\lambda)}{t} - \mathsf{negl}(\lambda). \tag{43}$$

From $\mathcal{G}'_i$ to $\mathcal{G}''_i$. Next, note that the machines $\mathcal{G}'_i$ and $\mathcal{G}''_i$ only differ in that $\mathcal{G}''_i$ no longer invokes $\mathcal{SE}_{\mathsf{ExtCom}}(1^\lambda, 1^{\varepsilon_1^{-1}})$ to extract from Step 2c on the left (see Algo. 6.3). As a consequence, it holds that

$$\forall i \in [\widetilde{t}],\ \Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}''_i(1^\lambda, \mathsf{pref}')\Big] \geq \frac{p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1] - 2\varepsilon_1(\lambda) - \varepsilon'(\lambda)}{t} - \varepsilon_1 - \mathsf{negl}(\lambda). \tag{44}$$

From $\mathcal{G}''_i$ to $\mathcal{G}''_1$. We first note that to finish our current proof of Lem. 16, it suffices to show the following inequality

$$\begin{aligned}\forall i \in [\widetilde{t}],\ \Big|\Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}''_i(1^\lambda, \mathsf{pref}')\Big] \\ - \Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}''_1(1^\lambda, \mathsf{pref}')\Big]\Big| \leq \mathsf{negl}(\lambda),\end{aligned} \tag{45}$$

because Inequalities (44) and (45) together imply Lem. 16 immediately. Thus, the only thing left is to prove Inequality (45).

Proof of Inequality (45). For the sake of contradiction, assume that there exist an $i \in [\widetilde{t}]$ and an inverse polynomial $\kappa(\lambda)$ such that for infinitely many $\lambda \in \mathbb{N}$, it holds that

$$\Big|\Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i''(1^\lambda, \mathsf{pref}')\Big] - \Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\Big]\Big| > \kappa(\lambda) \tag{46}$$

We next introduce a new machine $\widehat{\mathcal{G}}_i$ for this proof.

Algorithm 6.5: Machine $\widehat{\mathcal{G}}_i(1^\lambda, 1^{\widehat{\varepsilon}^{-1}}, \mathsf{pref}')$

Machine $\widehat{\mathcal{G}}_i(1^\lambda, 1^{\widehat{\varepsilon}^{-1}}, \mathsf{pref}')$: For each $i \in [\widetilde{t}]$, this machine works similar to $\mathcal{G}_i''(1^\lambda, \mathsf{pref}')$ (see Algo. 6.3), except that

- In Step 3 of the right session, instead of using the honest receiver's algorithm, $\widehat{\mathcal{G}}_i$ invokes $\mathcal{SE}_{\mathsf{ExtCom}}(1^\lambda, 1^{\widehat{\varepsilon}^{-1}})$ to extract a value $\widetilde{v}$, which is supposed to be the value committed by $\mathcal{M}_\lambda$ in the right Step 3.
  In more detail, the shares $\{\widetilde{\mathsf{cv}}_i^{(2)}\}_{i\in[n]}$ are committed by $\mathcal{M}_\lambda$ using independent ExtCom in parallel in the right Step 3. $\mathcal{G}_1$ will extract all of these shares using the parallel extractability of ExtCom with error parameter $\widehat{\varepsilon}$ (as per Def. 4), and compute $\widetilde{v} \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\widetilde{\mathsf{cv}}_1^{(2)}, \ldots, \widetilde{\mathsf{cv}}_n^{(2)})$.

We start by comparing the value $\widetilde{\alpha}$ committed to in Step 3 on the right in $\mathcal{G}_i''$ and the value $\widetilde{v}$ extracted by $\mathcal{SE}_{\mathsf{ExtCom}}$ in Step 3 on the right within $\widehat{\mathcal{G}}_i$. Similar to before, we can base this comparison on the simulation-extraction guarantee of $\mathcal{SE}_{\mathsf{ExtCom}}(1^\lambda, 1^{\widehat{\varepsilon}^{-1}})$, which implies that for any noticeable $\widehat{\varepsilon}$, it holds that

$$\forall i, j \in [\widetilde{t}], \Big|\Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_j\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i''(1^\lambda, \mathsf{pref}')\Big] - \Pr\Big[\big(\widetilde{v} = \widetilde{x}_j\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \widehat{\mathcal{G}}_i(1^\lambda, 1^{\widehat{\varepsilon}^{-1}}, \mathsf{pref}')\Big]\Big| \leq \widehat{\varepsilon}(\lambda). \tag{47}$$

Next, using a similar (non-uniform) argument as for Inequality (31) over the commit-and-prove protocol consisting of the hard puzzle setup step (i.e., Step 2) of the right session, we can establish the following: for any noticeable $\widehat{\varepsilon}(\lambda)$, it holds that

$$\forall i, j \in [\widetilde{t}], \Big|\Pr\Big[\big(\widetilde{v} = \widetilde{x}_j\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \widehat{\mathcal{G}}_i(1^\lambda, 1^{\widehat{\varepsilon}^{-1}}, \mathsf{pref}')\Big] - \Pr\Big[\big(\widetilde{v} = \widetilde{x}_j\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \widehat{\mathcal{G}}_1(1^\lambda, 1^{\widehat{\varepsilon}^{-1}}, \mathsf{pref}')\Big]\Big| \leq \mathsf{negl}(\lambda). \tag{48}$$

It then follows from Inequality (48) and Inequality (47) that for any noticeable $\widehat{\varepsilon}(\lambda)$, it holds that

$$\forall i, j \in [\widetilde{t}], \Big|\Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_j\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i''(1^\lambda, \mathsf{pref}')\Big] - \Pr\Big[\big(\widetilde{v} = \widetilde{x}_j\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \widehat{\mathcal{G}}_1(1^\lambda, 1^{\widehat{\varepsilon}^{-1}}, \mathsf{pref}')\Big]\Big| \leq \widehat{\varepsilon}(\lambda) + \mathsf{negl}(\lambda). \tag{49}$$

Setting $j = i$ in Inequality (49) implies that for any noticeable $\widehat{\varepsilon}(\lambda)$, it holds that

$$\forall i \in [\widetilde{t}], \Big|\Pr\Big[\big(\widetilde{\alpha} = \widetilde{x}_i\big) \wedge (\widetilde{d} = \top) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i''(1^\lambda, \mathsf{pref}')\Big]$$

$$- \Pr\left[\left(\widetilde{v} = \widetilde{x}_i\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \widehat{\mathcal{G}}_1(1^\lambda, 1^{\widehat{\varepsilon}^{-1}}, \mathsf{pref}')\right]\Bigg| \leq \widehat{\varepsilon}(\lambda) + \mathsf{negl}(\lambda). \quad (50)$$

Setting $i = 1$ (and then renaming j to i) in Inequality (49) implies that for any noticeable $\widehat{\varepsilon}(\lambda)$, it holds that

$$\forall i \in [\widetilde{t}], \Bigg| \Pr\left[\left(\widetilde{\alpha} = \widetilde{x}_i\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\right]$$
$$- \Pr\left[\left(\widetilde{v} = \widetilde{x}_i\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \widehat{\mathcal{G}}_1(1^\lambda, 1^{\widehat{\varepsilon}^{-1}}, \mathsf{pref}')\right]\Bigg| \leq \widehat{\varepsilon}(\lambda) + \mathsf{negl}(\lambda). \quad (51)$$

Combining Inequalities (50) and (51) implies that for any noticeable $\widehat{\varepsilon}(\lambda)$, it holds that

$$\forall i \in [\widetilde{t}], \Bigg| \Pr\left[\left(\widetilde{\alpha} = \widetilde{x}_i\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_i''(1^\lambda, \mathsf{pref}')\right]$$
$$- \Pr\left[\left(\widetilde{\alpha} = \widetilde{x}_i\right) \wedge \left(\widetilde{d} = \top\right) : (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1''(1^\lambda, \mathsf{pref}')\right]\Bigg| \leq 2 \cdot \widehat{\varepsilon}(\lambda) + \mathsf{negl}(\lambda). \quad (52)$$

By setting $\widehat{\varepsilon}(\lambda) := \frac{\kappa(\lambda)}{4}$ in Inequality (52), we obtain a contradiction to Inequality (46).

This concludes the proof of Inequality (45).

□

This eventually concludes our proof for Lem. 16. .

7 Simulation-Extractor $\mathcal{SE}$: 1-1 Settings

7.1 Noisy Simulation-Extraction Lemma

Lemma 17 (Noisy Simulatable-Extraction Lemma). *Let $\mathcal{G}$ be a QPT algorithm that takes the security parameter 1^λ, an error parameter $1^{\gamma^{-1}}$, a quantum state ρ, and a classical string z as input, and outputs $d \in \{\top, \bot\}$ and a quantum state ρ_{out}.*

Suppose that there exists a QPT algorithm $\mathcal{K}$ (referred to as the simulation-less extractor) that takes as input the security parameter 1^λ, two error parameters $1^{\gamma^{-1}}$ and $1^{\zeta^{-1}}$, a quantum state ρ, and a classical string z, and outputs $s \in \{0,1\}^{\mathsf{poly}(\lambda)} \cup \{\bot\}$ satisfying the following w.r.t. some sequence of classical strings $\{s_z^*\}_{z \in \{0,1\}^*}$.

1. *For any λ, ρ_λ, z_λ, and any noticeable functions $\gamma(\lambda)$ and $\zeta(\lambda)$, it holds that*

$$\Pr\left[s \notin \{s_{z_\lambda}^*, \bot\} \ : \ s \leftarrow \mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho_\lambda, z_\lambda)\right] \leq \zeta(\lambda) + \mathsf{negl}(\lambda).$$

2. *For any noticeable function $\gamma(\lambda)$, there exists a noticeable function $\delta(\lambda)$, which is efficiently computable from $\gamma(\lambda)$, so that the following requirement is satisfied: For any noticeable function $\zeta(\lambda)$ and any sequence $\{\rho_\lambda, z_\lambda\}_{\lambda \in \mathbb{N}}$ of polynomial-size quantum states and classical strings, if*

$$\Pr\left[d = \top \ : \ (d, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^\lambda, 1^{\gamma^{-1}}, \rho_\lambda, z_\lambda)\right] \geq 8\gamma(\lambda),$$

then

$$\Pr\left[s = s_{z_\lambda}^* \ : \ s \leftarrow \mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho_\lambda, z_\lambda)\right] \geq \delta(\lambda) - \zeta(\lambda) - \mathsf{negl}(\lambda).$$

Then, there exists a QPT algorithm $\mathcal{SE}$ such that for any noticeable function $\varepsilon = \varepsilon(\lambda)$, there exists a noticeable function $\gamma = \gamma(\lambda) \leq \varepsilon(\lambda)$ that is efficiently computable from ε and satisfies the following: For any sequence $\{\rho_\lambda, z_\lambda\}_{\lambda\in\mathbb{N}}$ of polynomial-size quantum states and classical strings,

$$\{\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \rho_\lambda, z_\lambda)\}_{\lambda\in\mathbb{N}} \stackrel{s}{\approx}_\varepsilon \{(\rho_{\mathsf{out}}, \Gamma_d(s^*_{z_\lambda})) \ : \ (d, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^\lambda, 1^{\gamma^{-1}}, \rho_\lambda, z_\lambda)\}_{\lambda\in\mathbb{N}},$$

where $\Gamma_d(s^*_{z_\lambda}) := \begin{cases} s^*_{z_\lambda} & \text{if } d = \top \\ \bot & \text{otherwise} \end{cases}$.

Proof sketch. Since the proof is almost identical to that of [LPY23b, Lemma 20], we only describe the differences.[31] There are the following two differences in the statement:

– We introduce an additional error parameter ζ, which gives an upper bound of the probability that $\mathcal{K}$ outputs $s \notin \{s^*_{z_\lambda}, \bot\}$. In [LPY23b, Lemma 20], the probability was assumed to be 0.

– The lower bound of $\mathcal{G}'s$ success probability in Item 2 is $8\gamma(\lambda)$ instead of $\gamma(\lambda)$.

The second point can be easily dealt with by simply replacing γ with 8γ in the original proof. The first point introduces an additional noticeable error polynomially related to ζ in the simulation for the case of $b = \top$. Since ζ can be chosen to be an arbitrarily small noticeable function, we can manage the additional error by appropriately setting the parameters.

Below, we give more concrete explanation for the readers who are familiar with the proof of [LPY23b, Lemma 20]. We only need to modify the proof of [LPY23b, Lemma 26], which claims that the simulation for the case $b = \top$ works. The first difference causes an error probability $\zeta + \mathsf{negl}(\lambda)$ in [LPY23b, Claims 29 and 30], which eventually causes an error $\zeta^{1/2} + \mathsf{negl}(\lambda)$ in [LPY23b, Eq. (84)] where the square root appears due to the gentle measurement lemma. As a result, [LPY23b, Eq. (84)] should be replaced with $\left(12(8\gamma)^{1/2} + 2\nu^{1/2}\right)^{1/2} + \zeta^{1/2} + \mathsf{negl}(\lambda)$ instead of $\left(12\gamma^{1/2} + 2\nu^{1/2}\right)^{1/2}$. (Note that γ is replaced with 8γ to deal with the second point as explained above.) It suffices to set $\gamma := \frac{1}{8}\left(\frac{\varepsilon}{10}\right)^4$, $\nu := \left(\frac{\varepsilon}{4}\right)^4$, and $\zeta := \left(\frac{\varepsilon}{2}\right)^2$ so that $\left(12(8\gamma)^{1/2} + 2\nu^{1/2}\right)^{1/2} + \zeta^{1/2} < \varepsilon$. □

7.2 Converting $\mathcal{K}$ to $\mathcal{SE}$

In this part, we build the simulation-extractor $\mathcal{SE}$ as required by Lem. 9. This eventually finishes the proof of Lem. 9, which is the only left piece in the proof of 1-1 non-malleability of Prot. 3.

The existence of the desired $\mathcal{SE}$ relies on Lem. 11 and 17 that we established previously. Roughly speaking, we will use Lem. 17 to convert the simulation-less extractor $\mathcal{K}$ from Lem. 11 to the desired $\mathcal{SE}$ satisfying the stipulated property in Lem. 9. However, we remark that this must be done with proper choice of the parameters. In the following, we show how this can be done.

We first define machines $\mathcal{K}'$ and $\mathcal{G}'$, which are "wrappers" for the machines $\mathcal{K}$ and $\mathcal{G}_1$ in Lem. 11. These machines will help us set parameters properly so we can invoke Lem. 17:

Machine $\mathcal{G}'$: it takes as input $(1^\lambda, 1^{\gamma^{-1}}, \mathsf{pref}')$ and proceeds as follows:

1. Set $\varepsilon := 8\gamma$.
2. Compute ε_1 from ε. Note that this can be done because Lem. 11 stipulates the there is a noticeable $\varepsilon_1 \leq \varepsilon$ that is efficiently computable from ε.
3. Run machine $\mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$ (as per Lem. 11) and output whatever it outputs.

Machine $\mathcal{K}'$: it takes as input $(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \mathsf{pref}')$ and proceeds as follows:

[31] [LPY23b] is the full version of [LPY23a] on arXiv.

1. Set $\varepsilon := 8\gamma$.
2. Compute ε_1 from ε. Note that this can be done because Lem. 11 stipulates the there is a noticeable $\varepsilon_1 \leq \varepsilon$ that is efficiently computable from ε.
3. Set $\varepsilon_2 := \zeta$.
4. Run machine $\mathcal{K}(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$ (as per Lem. 11) and output whatever it outputs.

In the following, we invoke Lem. 17 with $\mathcal{G}'$, $\mathcal{K}'$, $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \eta, \mathsf{VI}_\eta)$, $\widetilde{\tau}$, and $\mathsf{val}(\widetilde{\tau})$ playing the role of $\mathcal{G}$, $\mathcal{K}$, ρ_λ, z_λ, and $s^*_{z_\lambda}$ respectively in Lem. 17. To do that, we first prove the $\mathcal{G}'$ and $\mathcal{K}'$ indeed satisfy the conditions Items 1 and 2 in Lem. 17.

For Item 1 in Lem. 17. First, by Lem. 11, we know that the machine $\mathcal{K}$ when invoked with parameters $(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')$ outputs $\mathsf{val}(\widetilde{\tau})$ with probability at most $\varepsilon_2(\lambda) + \mathsf{negl}(\lambda)$. Since ε_2 is set to ζ in machine $\mathcal{K}'$, this implies that $\mathcal{K}'$ outputs $\mathsf{val}(\widetilde{\tau})$ with probability at most $\zeta(\lambda) + \mathsf{negl}(\lambda)$, satisfying Item 1 in Lem. 17.

For Item 2 in Lem. 17. First, we claim that if $\mathcal{G}'(1^\lambda, 1^{\gamma^{-1}}, \mathsf{pref}')$ output $\widetilde{d} = \top$ with probability 8γ, then it must hold that $p^{\mathsf{Sim}}_{\mathsf{pref}}[\varepsilon_1] \geq \varepsilon$, with the ε_1 defined in $\mathcal{G}'$. To see that, first notice that

$$\Pr\left[\widetilde{d} = \top \ : \ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}'(1^\lambda, 1^{\gamma^{-1}}, \mathsf{pref}')\right] = \Pr\left[\widetilde{d} = \top \ : \ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right]$$

where ε_1 is defined in the description of $\mathcal{G}'$. Thus, if the LHS of the above equation is greater than 8γ, then it must hold that

$$p^{\mathsf{Sim}}_{\mathsf{pref}}[\varepsilon_1] = \Pr\left[\widetilde{d} = \top \ : \ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')\right] \geq 8\gamma = \varepsilon.$$

Next, recall from Lem. 11 that under the condition of $p^{\mathsf{Sim}}_{\mathsf{pref}}[\varepsilon_1] \geq \varepsilon$, it must hold that

$$\Pr\left[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right] \geq \frac{\varepsilon'(\lambda) - \varepsilon_2(\lambda)}{\widetilde{t}},$$

where $\varepsilon' = \frac{\varepsilon(\lambda)}{10t^2}$. Then, by definition of $\mathcal{K}'$, it must hold that

$$\begin{aligned}
\Pr\left[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}'(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \mathsf{pref}')\right] &= \Pr\left[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}')\right] \\
&\geq \frac{\varepsilon'(\lambda) - \varepsilon_2(\lambda)}{\widetilde{t}} \\
&= \delta(\lambda) - \frac{\zeta(\lambda)}{\widetilde{t}} \\
&\geq \delta(\lambda) - \zeta(\lambda) - \mathsf{negl}(\lambda),
\end{aligned}$$

where we $\delta(\lambda) := \frac{\varepsilon'(\lambda)}{\widetilde{t}}$ with $\varepsilon' = \frac{\varepsilon(\lambda)}{10t^2}$. (Also note that $\varepsilon_2 = \zeta$ by definition of $\mathcal{K}'$.)

The above shows that the Item 2 in Lem. 17 is satisfied.

Invoking Lem. 17. Since $\mathcal{K}'$ and $\mathcal{G}'$ satisfy the conditions in Lem. 17, we can invoke it to claim the existence of a machine $\mathcal{SE}$ such that for any noticeable $\varepsilon(\lambda)$, there exists a noticeable $\gamma(\lambda) \leq \varepsilon(\lambda)$ such that

$$\{\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}')\}_{\lambda\in\mathbb{N}} \stackrel{\mathrm{s}}{\approx}_\varepsilon \{(\mathsf{OUT}, \mathsf{val}_{\widetilde{d}}(\widetilde{\tau})) \ : \ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}'(1^\lambda, 1^{\gamma^{-1}}, \mathsf{pref}')\}_{\lambda\in\mathbb{N}}.$$

Finally, recall that $\mathcal{G}'(1^\lambda, 1^{\gamma^{-1}}, \mathsf{pref}')$ is identical to machine $\mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}')$ where $\varepsilon_1 \leq \varepsilon(=8\gamma)$ is efficiently computable from γ as in the description of $\mathcal{G}'$. Thus, the above implies that: For any noticeable $\varepsilon(\lambda)$, there exists a noticeable $\varepsilon'(\lambda) \leq 8\varepsilon(\lambda)$ that is efficiently computable from $\varepsilon(\lambda)$, such that

$$\{\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}')\}_{\lambda\in\mathbb{N}} \stackrel{\mathrm{s}}{\approx}_\varepsilon \{(\mathsf{OUT}, \mathsf{val}_{\widetilde{d}}(\widetilde{\tau})) \ : \ (\mathsf{OUT}, \widetilde{d}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon'^{-1}}, \mathsf{pref}')\}_{\lambda\in\mathbb{N}},$$

which is exactly Lem. 9.

8 Post-Quantum Non-Malleable Commitments: One-to-One and Two-sided

In this section, we show how to remove the 'one-sided' restriction from Prot. 3.

Recall that our proof for the non-malleability of Prot. 3 works only if $t < \widetilde{t}$. However, this is not guaranteed in the real main-in-the-middle attack—the adversary can of course use a smaller tag in the right session. Fortunately, this problem can be addressed by the so-called 'two-slot' technique proposed by Pass and Rosen [PR05]. The idea is to create a situation where no matter how the MIM adversary $\mathcal{M}$ schedules the messages, there is always a 'slot' for which the '$t < \widetilde{t}$' condition holds; As long as this is true, non-malleability can be proven using the same techniques as we did for Prot. 3.

To do that, first observe that the only place where Prot. 3 makes use of the tag t is Step 2 **Hard Puzzle Setup**: The receiver is required to setup a t-solution hard puzzle where t is determined by the tag. Of course, Steps 4 and 5 also depend on t but that is rather a consequence of Step 2 using a t-solution hard puzzle.

This observation allows us to instantiate the [PR05] technique for Prot. 3 as follows. We view Step 2 as a 'slot' in [PR05] terminology. We ask the receiver to repeat this 'slot' twice sequentially, using t and $(T-t)$ as their respective tag, where recall that T is the upper-bound for the size of tag space and is a polynomial on the security parameter λ. That is,

- **Slot-A:** R first executes Step 2 as it is, setting a t-solution hard puzzle;
- **Slot-B:** R then executes Step 2 again, but using $(T-t)$ in place of t in the first execution. This sets a $(T-t)$-solution hard puzzle.

We also modify Steps 4 and 5 as follows:

- In Step 4, R reveals the solutions to *both* the t solutions w.r.t. **Slot-A** *and* the $(T-t)$ solutions w.r.t. **Slot-B**;
- In Step 5, we change the trapdoor statement from 'C manages to commit to a puzzle solution in Step 3' to 'C manages to commit to a puzzle solution *either* for **Slot-A** *or* for **Slot-B** in Step 3'.

By the above design, it is easy to see that one of the following case must happen no matter how $\mathcal{M}$ sets the tags t and $\widetilde{t}$:

1. $t = \widetilde{t}$: This is the trivial case that is already ruled out by the definition of non-malleability.
2. $t < \widetilde{t}$: In this case, non-malleability follows by applying the same argument as we did for Prot. 3 to **Slot-A**.
3. $t > \widetilde{t}$: In this case, it must hold that $(T-t) < (T-\widetilde{t})$. In other words, the tag for the left **Slot-B** is smaller than the tag for the right **Slot-B**. Therefore, non-malleability follows by applying the same argument as we did for Prot. 3 to **Slot-B**.

Therefore, the modified protocol is non-malleable without the 'one-sided' restriction.

We remark that the same technique has been employed by [LPY23a] to remove the 'one-sided' restriction in their original protocol as well. Our application does not encounter any new challenges compared with the same step in [LPY23a]. Thus, we omit the proof details and only present the formal description of this updated protocol in Appx. A.

We summarize the result of this section as the following theorem.

Theorem 10. *Assuming the existence of post-quantum one-way functions, there exists (i.e., Prot. 10) a black-box, constant-round construction of 1-1 (two-sided) post-quantum non-malleable commitments (as per Def. 5 with $k = 1$) in the synchronous setting, supporting tag space $[T]$ with $T(\lambda)$ being any polynomial in the security parameter λ.*

9 Simultaneous Extraction Lemma

So far, we have obtained a post-quantum non-malleable commitment in the 1-1 MIM setting. Recall that our final goal is to obtain a construction secure in the more demanding 1-many MIM setting. Jupping ahead, we will manage to show (in Sec. 10) that the same protocol Prot. 3 (more accurately, its two-sided version Prot. 10), without any modifications, is indeed already secure in the 1-many MIM setting. However, this or course requires a different security proof.

In this section, our develop a 'simultaneous extraction lemma' (Lem. 18). This lemma will play a crucial role later when we upgrade the security proof for Prot. 3 (or Prot. 10) to the 1-many setting in Sec. 10.

9.1 Lemma Statement

Lemma 18 (Simultaneous Extraction Lemma). *Let $\mathcal{V}$ be a QPT algorithm that takes the security parameter 1^λ, an error parameter $1^{\gamma^{-1}}$, a quantum state ρ, and a classical string z as input, and outputs $d \in \{\top, \bot\}$.*

Suppose that for $i \in [n]$, there exists a QPT algorithm $\mathcal{K}_i$ (referred to as the extractor) that takes as input the security parameter 1^λ, two error parameters $1^{\gamma^{-1}}$ and $1^{\zeta^{-1}}$, a quantum state ρ and outputs $s \in \{0,1\}^{\mathsf{poly}(\lambda)} \cup \{\bot\}$ satisfying the following w.r.t. some sequence of classical strings $\{s^*_{z,i}\}_{z\in\{0,1\}^*, i\in[n]}$.

- **Assumption 1:** *For any λ, ρ_λ, z_λ, $i \in [n]$, and any noticeable functions $\gamma(\lambda)$ and $\zeta(\lambda)$, it holds that*

$$\Pr\left[s \notin \{s^*_{z_\lambda,i}, \bot\} \ : \ s \leftarrow \mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho_\lambda, z_\lambda)\right] \leq \zeta(\lambda) + \mathsf{negl}(\lambda).$$

- **Assumption 2:** *For any noticeable function $\gamma(\lambda)$, there exists a noticeable function $\delta(\lambda)$, which is efficiently computable from $\gamma(\lambda)$, so that the following requirement is satisfied: For any noticeable function $\zeta(\lambda)$ and any sequence $\{\rho_\lambda, z_\lambda\}_{\lambda\in\mathbb{N}}$ of polynomial-size quantum states and classical strings and $i \in [n]$, if*

$$\Pr\left[d = \top \ : \ d \leftarrow \mathcal{V}(1^\lambda, 1^{\gamma^{-1}}, \rho_\lambda, z_\lambda)\right] \geq \gamma(\lambda),$$

then

$$\Pr\left[s = s^*_{z_\lambda,i} \ : \ s \leftarrow \mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho_\lambda, z_\lambda)\right] \geq \delta(\lambda) - \zeta(\lambda) - \mathsf{negl}(\lambda).$$

Then, there exists a QPT algorithm $\mathcal{K}$ that satisfies the following:

1. *For any λ, ρ_λ, z_λ, and any noticeable functions $\gamma(\lambda)$ and $\zeta(\lambda)$, it holds that*

$$\Pr\left[\overline{s} \notin \{s^*_{z_\lambda,1}||s^*_{z_\lambda,2}...||s^*_{z_\lambda,n}, \bot\} \ : \ \overline{s} \leftarrow \mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho_\lambda, z_\lambda)\right] \leq \zeta(\lambda) + \mathsf{negl}(\lambda).$$

2. *For any noticeable function $\gamma(\lambda)$, there exists a noticeable function $\delta'(\lambda)$, which is efficiently computable from $\gamma(\lambda)$, so that the following requirement is satisfied: For any noticeable funntion $\zeta(\lambda)$ and any sequence $\{\rho_\lambda, z_\lambda\}_{\lambda\in\mathbb{N}}$ of polynomial-size quantum states and classical strings, if*

$$\Pr\left[d = \top \ : \ d \leftarrow \mathcal{V}(1^\lambda, 1^{\gamma^{-1}}, \rho_\lambda, z_\lambda)\right] \geq 8\gamma(\lambda),$$

then

$$\Pr\left[\overline{s} = s^*_{z_\lambda,1}||s^*_{z_\lambda,2}...||s^*_{z_\lambda,n} \ : \ \overline{s} \leftarrow \mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho_\lambda, z_\lambda)\right] \geq \delta'(\lambda) - \zeta(\lambda) - \mathsf{negl}(\lambda).$$

9.2 Preparation

We take several tools and lemmas from [Zha20, CMSZ21] and give slight extensions of them.

Definition 15 (Projective Implementation [Zha20]). *Let $\mathcal{M} = (M_0, M_1)$ be a binary outcome POVM. Let $\mathcal{E} = \{E_p\}_{p\in S}$ be a projective measurement indexed by $p \in S$ for some finite subset S of $[0,1]$.*[32] *Consider the following experiment:*

1. *Apply the measurement $\mathcal{E}$ to obtain $p \in S$.*
2. *Output 1 with probability p and output 0 with probability $1-p$.*

We say that $\mathcal{E}$ is a projective implementation of $\mathcal{M}$ if for any initial state, the above experiment yields the identical distribution to that obtained by applying the POVM $\mathcal{M}$.

Lemma 19 ([Zha20, Lemma 3.3]). *Any binary outcome POVM $\mathcal{M}$ has a unique projective implementation.*

For a binary outcome POVM $\mathcal{M}$, we write $\mathsf{ProjImp}(\mathcal{M})$ to mean its projective implementation.

Definition 16 (Shift Distance [Zha20]). *For two distributions D_0, D_1, with cumulative density functions f_0, f_1, respectively, the shift distance with parameter ε is defined as*

$$\Delta^\varepsilon_{\mathsf{Shift}}(D_0, D_1) := \sup_{x\in\mathbb{R}} \min_{y\in[f_1(x-\varepsilon), f_1(x+\varepsilon)]} |f_0(x) - y|.$$

For two real-valued measurements $\mathcal{M}$ and $\mathcal{N}$ over the same quantum system, the shift distance between $\mathcal{M}$ and $\mathcal{N}$ with parameter ε is

$$\Delta^\varepsilon_{\mathsf{Shift}}(\mathcal{M}, \mathcal{N}) := \sup_{|\psi\rangle} \Delta^\varepsilon_{\mathsf{Shift}}(\mathcal{M}(|\psi\rangle), \mathcal{N}(|\psi\rangle)).$$

By the definition, we can see the following: If $\Delta^\varepsilon_{\mathsf{Shift}}(\mathcal{M}, \mathcal{N}) \leq \eta$, then for any state $|\psi\rangle$ and $x \in \mathbb{R}$,

$$\Pr[\mathcal{M}(|\psi\rangle) \leq x] \leq \Pr[\mathcal{N}(|\psi\rangle) \leq x + \varepsilon] + \eta, \qquad \Pr[\mathcal{M}(|\psi\rangle) \geq x] \leq \Pr[\mathcal{N}(|\psi\rangle) \geq x - \varepsilon] + \eta,$$
$$\Pr[\mathcal{N}(|\psi\rangle) \leq x] \leq \Pr[\mathcal{M}(|\psi\rangle \leq x + \varepsilon] + \eta, \qquad \Pr[\mathcal{N}(|\psi\rangle) \geq x] \leq \Pr[\mathcal{M}(|\psi\rangle \geq x - \varepsilon] + \eta.$$

Definition 17 (Almost Projective Measurements [Zha20]). *A real-valued measurement $\mathcal{M} = (M_i)_{i\in I}$ is (ε, η)-almost projective if the following is true: for any quantum state $|\psi\rangle$, apply $\mathcal{M}$ twice in a row to $|\psi\rangle$, obtaining outcomes x, y. Then $\Pr[|x - y| \leq \varepsilon] \geq 1 - \eta$.*

[32] In [Zha20], $\mathcal{E}$ is labeled by a distribution D. The definition here is identical to theirs if we interpret p as a distribution that takes 1 with probability p and otherwise takes 0.

The following is a variant of [Zha20, Theorem 6.2].

Lemma 20. *For any binary-outcome POVM $\mathcal{M} = (M_0, M_1)$ and reals $0 < \varepsilon, \eta < 1$, there is a real-valued measurement $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}}$ that satisfies the following:*

1. $\Delta^{\varepsilon}_{\mathsf{Shift}}(\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}}, \mathsf{ProjImp}(\mathcal{M})) \leq \eta$.
2. $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}}$ *is* (ε, η)*-almost projective.*
3. *The run time of* $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}}$ *is* $T_{\mathcal{M}} \cdot \mathsf{poly}(\varepsilon^{-1}, \log(\eta^{-1}))$*, where* $T_{\mathcal{M}}$ *is the run time of the POVM* $\mathcal{M}$*.*

There are the following two differences from the original statement of [Zha20, Theorem 6.2].

1. We consider general binary-outcome POVM whereas they focuses on a special case called "mixture of projective measurements."
2. We require the run time of $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}}$ is $T_{\mathcal{M}} \cdot \mathsf{poly}(\varepsilon^{-1}, \log(\eta^{-1}))$ whereas they require it only for the *expected* run time.

For the first difference, we observe that the original proof can be easily extended to general binary-outcome POVM by using Jordan's lemma. The second difference can be resolved by using an idea of "scaling down" as sketched in [Zha20, Remark 6.4].

For completeness, we prove Lem. 20. We note that the proof is based on the proof of [Zha20, Theorem 6.2] and we often repeat very similar arguments to theirs.

Proof of Lem. 20. First, we construct $\widetilde{\mathsf{API}}^{\varepsilon,\eta}_{\mathcal{M}}$ that satisfies the requirements if $\mathsf{ProjImp}(\mathcal{M})$ is supported by $p \in [1/4, 3/4]$, i.e., for any state ρ, it holds that

$$\Pr\left[\frac{1}{4} \leq p \leq \frac{3}{4} \ : \ p \leftarrow \mathsf{ProjImp}(\mathcal{M})(\rho)\right] = 1.$$

Looking ahead, this assumption is used to make sure that $\widetilde{\mathsf{API}}^{\varepsilon,\eta}_{\mathcal{M}}$ runs in strict QPT (rather than expected QPT as in [Zha20, Theorem 6.2]). At the end of the proof, we modify it to $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}}$ that works for any binary-outcome measurement.

Suppose that $\mathsf{ProjImp}(\mathcal{M})$ is supported by $p \in [1/4, 3/4]$. Let $\mathbf{X}$ be a quantum register for states on which $\mathcal{M} = (M_0, M_1)$ acts. Let U be a purification of $\mathcal{M}$ on $\mathbf{X}$ and an ancilla register $\mathbf{Y}$. That is, we define the unitary U in such a way that for any state $\rho_{\mathbf{X}}$ on $\mathbf{X}$ and $b \in \{0, 1\}$, we have

$$\mathrm{Tr}(M_b \rho_{\mathbf{X}}) = \mathrm{Tr}\big(U^\dagger(|b\rangle\langle b| \otimes I)U(\rho_{\mathbf{X}} \otimes |0\rangle\langle 0|_{\mathbf{Y}})\big)$$

where $|b\rangle\langle b| \otimes I$ means the operator that projects the first qubit of $\mathbf{X}$ onto $|b\rangle$. We define two projectors Π_0 and Π_1 over $\mathbf{X}$ and $\mathbf{Y}$ as:

$$\Pi_0 := I_{\mathbf{X}} \otimes |0^n\rangle\langle 0^n|_{\mathbf{Y}}, \quad \Pi_1 := U^\dagger(|1\rangle\langle 1| \otimes I)U$$

where n is the number of qubits in $\mathbf{Y}$. By applying Jordan's lemma to Π_0 and Π_1, we can see that there is an orthogonal decomposition of the Hilbert space over $\mathbf{X}$ and $\mathbf{Y}$ into two-dimensional subspaces $\{S_j\}_j$ that satisfies the following:[33] For each two-dimensional subspace S_j, there exist two orthonormal bases $(|\alpha_j\rangle, |\alpha_j^\perp\rangle)$ and $(|\beta_j\rangle, |\beta_j^\perp\rangle)$ of S_j such that

$$\Pi_0 |\alpha_j\rangle = |\alpha_j\rangle, \quad \Pi_0 |\alpha_j^\perp\rangle = 0,$$

[33] In general, there may also appear one-dimensional subspaces. However, by our assumption that $\mathsf{ProjImp}(\mathcal{M})$ is supported by $p \in [1/4, 3/4]$, all eigenvalues of $\Pi_0\Pi_1\Pi_0$ belongs to $[1/4, 3/4]$, and thus one-dimensional subspaces do not appear in our case.

$$\Pi_1 \left|\beta_j\right\rangle = \left|\beta_j\right\rangle, \quad \Pi_1 \left|\beta_j^\perp\right\rangle = 0.$$

Moreover, if we let

$$p_j := \left\langle\alpha_j\right| \Pi_1 \left|\alpha_j\right\rangle,$$

then we have $1/4 \le p_i \le 3/4$ and

$$\left|\alpha_j\right\rangle = \sqrt{p_j}\left|\beta_j\right\rangle + \sqrt{1-p_j}\left|\beta_j^\perp\right\rangle, \quad \left|\beta_j\right\rangle = \sqrt{p_j}\left|\alpha_j\right\rangle + \sqrt{1-p_j}\left|\alpha_j^\perp\right\rangle.$$

In particular, this implies that

$$\begin{aligned}
\Pi_1 \left|\alpha_j\right\rangle &= \sqrt{p_j}\left|\beta_j\right\rangle, & (I-\Pi_1)\left|\alpha_j\right\rangle &= \sqrt{1-p_j}\left|\beta_j^\perp\right\rangle, \\
\Pi_1 \left|\alpha_j^\perp\right\rangle &= \sqrt{1-p_j}\left|\beta_j\right\rangle, & (I-\Pi_1)\left|\alpha_j^\perp\right\rangle &= \sqrt{p_j}\left|\beta_j^\perp\right\rangle, \\
\Pi_0 \left|\beta_j\right\rangle &= \sqrt{p_j}\left|\alpha_j\right\rangle, & (I-\Pi_0)\left|\beta_j\right\rangle &= \sqrt{1-p_j}\left|\alpha_j^\perp\right\rangle, \\
\Pi_0 \left|\beta_j^\perp\right\rangle &= \sqrt{1-p_j}\left|\alpha_j\right\rangle, & (I-\Pi_0)\left|\beta_j\right\rangle &= \sqrt{p_j}\left|\alpha_j^\perp\right\rangle
\end{aligned} \tag{53}$$

Since $\Pi_0 \left|\alpha_j\right\rangle = \left|\alpha_j\right\rangle$, we can write $\left|\alpha_j\right\rangle = \left|\alpha_j'\right\rangle_{\mathbf{X}} \left|0\right\rangle_{\mathbf{Y}}$ for each j. For each $p \in [1/4, 3/4]$, we define a projector E_p on $\mathbf{X}$ as

$$E_p := \sum_{j:p_j=p} \left|\alpha_j'\right\rangle \left\langle\alpha_j'\right|.$$

Then one can see that $\mathcal{E} = \{E_p\}_{p\in S}$ is the projective implementation of $\mathcal{M}$ where

$$S := \{p \in [1/4, 3/4] \ : \ \exists j \text{ s.t. } p_j = p\}.$$

We describe the algorithm $\widetilde{\mathsf{API}}_{\mathcal{M}}^{\varepsilon,\eta}$ on register $\mathbf{X}$ below:

1. Prepare and initialize the register $\mathbf{Y}$ to the all-zero state.
2. Initialize a classical list $L = (0)$.
3. Repeat the following "main loop" for $i = 1, 2, \ldots, T$, where $T := \lceil \ln(6/\eta)/\varepsilon^2 \rceil$:
   (a) Apply the projective measurement $(I - \Pi_1, \Pi_1)$, obtaining an outcome b_{2i-1}, and append b_{2i-1} to the end of L.
   (b) Apply the projective measurement $(\Pi_0, I - \Pi_0)$, obtaining an outcome b_{2i}, and append b_{2i} to the end of L.
4. Let t be the number of bit flips in the sequence $L = (0, b_1, b_2, ..., b_{2T})$, and let $\widetilde{p} := t/2T$.
5. If $b_{2T} = 1$, repeat the "main loop" until the first time $b_{2i} = 0$ or it is repeated $T' = \lceil \log_{5/8}(\eta/3) \rceil$ times. We say that it fails if $b_{2i} = 0$ does not occur within T' times repetition.
6. Discard $\mathbf{Y}$ and output $\widetilde{p}$.

The run time requirement of Item 3 is clear from the description. We next establish one by one. First, we remark that $\widetilde{\mathsf{API}}_{\mathcal{M}}^{\varepsilon,\eta}$ just applies projective measurements $(I - \Pi_1, \Pi_1)$ and $(\Pi_0, I - \Pi_0)$ on registers $\mathbf{X}, \mathbf{Y}$. Therefore, when proving Items 1 and 2, we can analyze each subspace separately.

That is, we can focus on the case where the initial state is $\left|\alpha_j'\right\rangle$ for some j.

Proving Item 1 of Lem. 20. Note that $\mathsf{ProjImp}(\mathcal{M})$ on $\left|\alpha_j'\right\rangle$ results in p_j with probability 1. By Eq. (53), we can see that the list L obtained by applying $\widetilde{\mathsf{API}}_{\mathcal{M}}^{\varepsilon,\eta}$ on $\left|\alpha_j'\right\rangle$ is according to the following distribution:

– Let K be a list of $2T$ independent coin flips with expected value p_j.

– Set L_i be the parity of the first i bits of K.

Then $t = 2T\widetilde{p}$ is the number of 1s in K. Thus, by Hoeffding's bound, we have

$$\Pr\left[|p_j - \widetilde{p}| \geq \varepsilon/2\right] \leq 2e^{-2(2T)(\varepsilon/2)^2} \leq \eta/3 < \eta$$

where we used $T \geq \ln(6/\eta)/\varepsilon^2$. This implies $\Delta^{\varepsilon}_{\mathsf{Shift}}(\widetilde{\mathsf{API}}^{\varepsilon,\eta}_{\mathcal{M}}, \mathsf{ProjImp}(\mathcal{M})) \leq \eta$, finishing the proof of Item 1.

Proving Item 2 of Lem. 20. Suppose that we sequentially run $\widetilde{\mathsf{API}}^{\varepsilon,\eta}_{\mathcal{M}}$ twice on the initial state $\left|\alpha'_j\right\rangle$. Let $\widetilde{p}_0$ and $\widetilde{p}_1$ be the measurement outcome of the first and second application, respectively. If the first application of $\widetilde{\mathsf{API}}^{\varepsilon,\eta}_{\mathcal{M}}$ does not fail, then the state in $\mathbf{X}$ goes back to $\left|\alpha'_j\right\rangle$ at the end of the first application. Thus, by repeating a similar analysis to the above, we can see that

$$\Pr\left[|\widetilde{p}_b - p_j| \geq \varepsilon/2\right] \leq \eta/3$$

for $b \in \{0,1\}$ conditioned on that the first application does not fail. Moreover, each trial in Step 5 of the description of $\widetilde{\mathsf{API}}^{\varepsilon,\eta}_{\mathcal{M}}$ succeeds with probability $2p_j(1-p_j) \geq 3/8$ where we used $p_j \in [1/4, 3/4]$, and thus the probability of failure is at most $(1-3/8)^{T'} \leq \eta/3$ where we used $T' \geq \log_{5/8}(\eta/3)$. Combining the above, we have

$$\Pr[|\widetilde{p}_0 - \widetilde{p}_1| \geq \varepsilon] \leq \eta/3 + \eta/3 + \eta/3 = \eta,$$

which implies Item 2. This finishes the proof of Lem. 20 for the case where $\mathsf{ProjImp}(\mathcal{M})$ is supported by $p \in [1/4, 3/4]$.

For the General Case of $p \in [0,1]$. Finally, we extend it to general binary-outcome POVMs. For any binary-outcome POVM $\mathcal{M} = (M_0, M_1)$, let $\mathcal{M}' := (\frac{I}{4} + \frac{M_0}{2}, \frac{I}{4} + \frac{M_1}{2})$. That is, $\mathcal{M}'$ corresponds to the process that either outputs a uniformly random bit or applies $\mathcal{M}$ with probability $1/2$ for each. Let $\mathcal{E} = \{E_p\}_{p \in S}$ be the projective implementation of $\mathcal{M}$. Then it is easy to see that the projective implementation of $\mathcal{M}'$ is $\mathcal{E}' = \{E'_{p'}\}_{p' \in S'}$ where $E'_{p'} := E_{2p'-1/2}$ and $S' := \{p' : 2p' - 1/2 \in S\}$. For any $p' \in S'$, since $2p' - 1/2 \in [0,1]$, we have $p' \in [1/4, 3/4]$. Thus, $\mathsf{ProjImp}(\mathcal{M}')$ is supported by $p' \in [1/4, 3/4]$ and $\widetilde{\mathsf{API}}$ is applicable for $\mathcal{M}'$. Based on this observation, we construct $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}}$ as follows:

1. Apply $\widetilde{\mathsf{API}}^{\varepsilon/2,\eta}_{\mathcal{M}'}$, obtaining an outcome p'.
2. Output $p := 2p' - 1/2$.

Then the properties of $\widetilde{\mathsf{API}}^{\varepsilon/2,\eta}_{\mathcal{M}'}$ which we showed above are directly translated into those of $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}}$, which concludes the proof of Lem. 20.

□

Lemma 21 ([CMSZ21, Lemma 4.10]). *Let $\mathcal{N}$ be an (ε,η)-almost projective measurement on a Hilbert space $\mathcal{H}$, and $\mathcal{P} = (P_0, P_1)$ be a binary-outcome projective measurement on $\mathcal{H}$. Then there is a quantum algorithm $\mathsf{Repair}^{\mathcal{N},\mathcal{P}}$ on $\mathcal{H}$ satisfying the following:*

– *For a positive integer T, consider the following procedure $\mathsf{RepairExpt}^{\mathcal{N},\mathcal{P}}(1^T)$ on $\mathcal{H}$:*

1. *Apply $\mathcal{N}$, obtaining outcome p;*
2. *Apply $\mathcal{P}$, obtaining outcome b;*

3. *Apply* $\mathsf{Repair}^{\mathcal{N},\mathcal{P}}(1^T,b,p)$.

4. *Output* p.

Then $\mathsf{RepairExpt}^{\mathcal{N},\mathcal{P}}(1^T)$ *is* $(2\varepsilon, 2(\eta+1/T)+4\sqrt{\eta})$*-almost projective.*

– *The expected run time of* $\mathsf{Repair}^{\mathcal{N},\mathcal{P}}(1^T,b,p)$ *is at most* $(T_{\mathcal{N}}+T_{\mathcal{P}})\cdot(4T\sqrt{\eta}+3)$ *where* $T_{\mathcal{N}}$ *and* $T_{\mathcal{P}}$ *are run times of* $\mathcal{N}$ *and* $\mathcal{P}$*, respectively.*

We show the following corollary.

Corollary 5. *Let* $\mathcal{N}$ *be an* (ε,η)*-almost projective measurement on a Hilbert space* $\mathcal{H}$*, and* $\mathcal{A}$ *be a quantum algorithm that takes a quantum state in* $\mathcal{H}$ *as input and outputs a classical string satisfying the following: There are some classical string* s^* *and* $0\le\zeta\le 1$ *such that for any state* ρ*,*

$$\Pr[s\notin\{s^*,\bot\}\ :\ s\leftarrow\mathcal{A}(\rho)]\le\zeta.$$

Then for any positive integer T *and* $p\in[0,1]$*, there is a measurement* $\mathcal{A}\text{-}\mathsf{Repair}(1^T,p)$ *satisfying the following:*

– *For any* T*,* p*, and any state* ρ *in* $\mathcal{H}$*, if we apply* $\mathcal{A}\text{-}\mathsf{Repair}(1^T,p)$ *on* ρ*, then the distribution of the measurement outcome is identical to that of* $\mathcal{A}(\rho)$*.*

– *For a positive integer* T*, consider the following procedure* $\mathsf{RepairExpt}^{\mathcal{N},\mathcal{A}\text{-}\mathsf{Repair}}(1^T)$ *on* $\mathcal{H}$*:*

1. *Apply* $\mathcal{N}$*, obtaining outcome* p*;*

2. *Apply* $\mathcal{A}\text{-}\mathsf{Repair}(1^T,p)$*, obtaining outcome* s*;*

3. *Output* p*.*

Then $\mathsf{RepairExpt}^{\mathcal{N},\mathcal{A}\text{-}\mathsf{Repair}}(1^T)$ *is* $\left(2\varepsilon, 2(\eta+1/T)+4\sqrt{\eta}+\sqrt{\zeta}\right)$*-almost projective.*

– *The expected run time of* $\mathcal{A}\text{-}\mathsf{Repair}(1^T,p)$ *is at most* $(T_{\mathcal{N}}+T_{\mathcal{A}})\cdot(4T\sqrt{\eta}+3)$ *where* $T_{\mathcal{N}}$ *and* $T_{\mathcal{A}}$ *are run times of* $\mathcal{N}$ *and* $\mathcal{A}$*, respectively.*

Proof. Intuitively, $\mathcal{A}\text{-}\mathsf{Repair}$ first runs $\mathcal{A}$ and then applies the repair procedure of Lem. 21. A formal proof is given below.

We can describe $\mathcal{A}$ by using a unitary U over the input register Inp, output register $\mathbf{S}$, and working register $\mathbf{W}$ as follows:

$\mathcal{A}(\rho)$: Set ρ in Inp, initialize $\mathbf{S}$ and $\mathbf{W}$ to be all-zero states, apply U, measure $\mathbf{S}$, and output the outcome s.

We define a binary projective measurement $\mathcal{P}=(P_0,P_1)$ on Inp, $\mathbf{S}$, and $\mathbf{W}$ as

$$P_1 := U^\dagger\Big(\sum_{s\neq\bot}|s\rangle\langle s|\Big)_{\mathbf{S}}U$$

and $P_0 := I-P_1$. We apply Lem. 21 for $\mathcal{N}$ and $\mathcal{P}$ to get $\mathsf{Repair}^{\mathcal{N},\mathcal{P}}$ satisfying the requirements of Lem. 21.[34] By using it, we construct $\mathcal{A}\text{-}\mathsf{Repair}(1^T,p)$ on Inp as follows:

1. Initialize $\mathbf{S}$ and $\mathbf{W}$ to all-zero states.

2. Apply $\mathcal{P}$, obtaining an outcome b.

[34] Strictly speaking, $\mathcal{N}$ is a POVM on Inp but $\mathcal{P}$ is a projector on $(\mathsf{Inp},\mathbf{S},\mathbf{W})$ and thus Lem. 21 is not directly applicable. We abuse the notation to simply write $\mathcal{N}$ to mean its trivial extension to registers $(\mathsf{Inp},\mathbf{S},\mathbf{W})$ that does not touch $(\mathbf{S},\mathbf{W})$.

3. If $b = 0$, then set $s := \bot$. If $b = 1$, then apply U, measure $\mathbf{S}$ to obtain s, and apply $U^\dagger$.
4. Apply $\mathsf{Repair}^{\mathcal{N},\mathcal{P}}(1^T, b, p)$.
5. Output s as the measurement outcome.

It is clear from the construction that the distribution of s obtained by applying $\mathcal{A}$-$\mathsf{Repair}(p)$ on ρ is identical to the distribution of $\mathcal{A}(\rho)$. The requirement about the run time directly follows from that of Lem. 21. Below, we show that $\mathsf{RepairExpt}^{\mathcal{N},\mathcal{A}\text{-}\mathsf{Repair}}(1^T)$ is $(2\varepsilon, 2(\eta+1/T)+4\sqrt{\eta}+\sqrt{\zeta})$-almost projective.

Let $\mathcal{A}$-$\mathsf{Repair}'(p)$ be a quantum process that works similarly to $\mathcal{A}$-$\mathsf{Repair}(p)$ except that Steps 3 and 5 are removed. Then, it is not hard to see that $\mathsf{RepairExpt}^{\mathcal{N},\mathcal{A}\text{-}\mathsf{Repair}'}(1^T)$ is identical to $\mathsf{RepairExpt}^{\mathcal{N},\mathcal{P}}(1^T)$, and thus it is $(2\varepsilon, 2(\eta+1/T)+4\sqrt{\eta})$-almost projective by Lem. 21. Moreover, we observe that the measurement in Step 3 of $\mathcal{A}$-$\mathsf{Repair}(p)$ for the case of $b = 1$ yields a fixed value s^* with probability except for ζ by the assumption about $\mathcal{A}$. Thus, by the gentle measurement lemma [Aar05, Lemma 2.2], the trace distance between the states before and after the step is at most $\sqrt{\zeta}$. This implies that $\mathsf{RepairExpt}^{\mathcal{N},\mathcal{A}\text{-}\mathsf{Repair}}(1^T)$ is $(2\varepsilon, 2(\eta+1/T)+4\sqrt{\eta}+\sqrt{\zeta})$-almost projective.

This finishes the proof of Corollary 5. □

9.3 Proof of Lem. 18

Let $\mathcal{M}_{\lambda,\gamma,z}$ be the binary-outcome POVM corresponding to $\mathcal{V}(1^\lambda, 1^{\gamma^{-1}}, \cdot, z)$. That is, it is defined in such a way that $\Pr\left[\mathcal{M}_{\lambda,\gamma,z}(\rho) = 1\right] = \Pr\left[\mathcal{V}(1^\lambda, 1^{\gamma^{-1}}, \rho, z) = \top\right]$ for any state ρ. Let $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}$ be the (ε,η)-almost projective measurement as given in Lem. 20. For each $i, \lambda, \gamma, \zeta, \rho, z, \varepsilon, \eta$, we apply Corollary 5 to the (ε,η)-almost projective measurement $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}$ and the algorithm $\mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \cdot, z)$, and we denote the corresponding repairing measurement by $\mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \cdot, z)$-$\mathsf{Repair}$ and the corresponding repairing experiment by $\mathsf{RepairExpt}^{\varepsilon,\eta}_{\lambda,\gamma,\zeta,z}(1^T)$.[35] By the assumption about $\mathcal{K}_i$ and Corollary 5, $\mathsf{RepairExpt}^{\varepsilon,\eta}_{\lambda,\gamma,\zeta,z}(1^T)$ is $\left(2\varepsilon, 2(\eta+1/T)+4\sqrt{\eta}+\sqrt{\zeta}+\mathsf{negl}(\lambda)\right)$-almost projective.

We first construct *expected* QPT algorithm $\mathcal{K}$ that satisfies the requirements, after which we argue that we can modify it to be *strict* QPT by truncation.

The *expected* QPT algorithm $\mathcal{K}$ is described as follows:

$\mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho, z)$: Do the following:

1. Compute δ as in **Assumption 2** of Lem. 18 from the given γ.
2. Take an positive integer T' in such a way that $(1-\delta/3)^{T'} \leq \gamma/n$ holds. (For example, $T' = O(\delta^{-1}\log\gamma^{-1}\log n)$ suffices).
3. Set parameters as follows:

$$\begin{aligned} T &:= \lceil 6nT'\gamma^{-1}\rceil \\ \varepsilon &:= \min\{2\gamma/(2nT'+1), \gamma/4\} \\ \eta &:= \left(\gamma/(18nT')\right)^2 \\ \zeta' &:= \min\{\zeta/(nT'), \left(\gamma/(3nT')\right)^2, \delta/2\} \end{aligned}$$

4. Apply $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}$ on Inp to obtain an outcome $\widetilde{p}$. If $\widetilde{p} < 4\gamma - \varepsilon$, output $\bot$ and halt.

[35] If we strictly follow the notation in Corollary 5, then the experiment should be written as $\mathsf{RepairExpt}^{\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}},\mathcal{K}_i(1^\lambda,1^{\gamma^{-1}},1^{\zeta^{-1}},\cdot,z)\text{-}\mathsf{Repair}}(1^T)$, but we simply write $\mathsf{RepairExpt}^{\varepsilon,\eta}_{\lambda,\gamma,\zeta,z}(1^T)$ for brevity.

5. For $i = 1, 2, ..., n$, do the following:
   (a) For $j = 1, 2, ..., T'$, do the following
      i. Apply $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}$ to obtain an outcome $\widetilde{p}_{i,j}$.
      ii. If $\widetilde{p}_{i,j} < \widetilde{p}_{i,j-1} - 2\varepsilon$, output $\bot$ and halt, where when $i = 1$ and $j = 1$, $\widetilde{p}_{i,j-1} := \widetilde{p}$ and when $i \geq 2$ and $j = 1$, $\widetilde{p}_{i,j-1} := \widetilde{p}_{i-1,T'}$.
      iii. Apply $\mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta'^{-1}}, \cdot, z)$-$\mathsf{Repair}^{\varepsilon,\eta}(1^T, \widetilde{p}_{i,j})$ to obtain an outcome $s_{i,j}$.
      iv. If $s_{i,j} \neq \bot$, set $s_i := s_{i,j}$, break the inner loop, and proceed to the outer loop for $i + 1$.
   (b) If $s_{i,j} = \bot$ for all $j \in [T']$, output $\bot$ and halt.
6. Output $s_1||s_2||...||s_n$.

We can see that $\mathcal{K}$ runs in expected QPT by Lem. 20 and Corollary 5.

Proving Property 1 of Lem. 18. We observe that whenever $\mathcal{K}$ does not output $\bot$, for each $i \in [n]$, s_i is a non-$\bot$ value obtained by $\mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta'^{-1}}, \cdot, z)$-$\mathsf{Repair}^{\varepsilon,\eta}(\widetilde{p}_{i,j})$. By Corollary 5, its distribution is identical to the output distribution of $\mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta'^{-1}}, \cdot, z)$.

By **Assumption 1** of Lem. 18, it outputs non-$\bot$ value other than $s^*_{z,i}$ with probability at most $\zeta' + \mathsf{negl}(\lambda)$. Since we apply it at most T' times, the probability that it ever occurs is at most $T'(\zeta' + \mathsf{negl}(\lambda))$. By taking union bound over all $i \in [n]$, the probability that it occurs for some $i \in [n]$ is at most $nT'(\zeta' + \mathsf{negl}(\lambda)) \leq \zeta + \mathsf{negl}(\lambda)$. This finishes the proof of Property 1.

Proving Property 2 of Lem. 18. Suppose that ρ and z satisfy the requirement of Property 2, i.e., we have

$$\Pr\left[d = \top \ : \ d \leftarrow \mathcal{V}(1^\lambda, 1^{\gamma^{-1}}, \rho, z)\right] \geq 8\gamma. \tag{54}$$

We define the following events in the execution of $\mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho, z)$:

- Bad_1: The event that $\mathcal{K}$ returns $\bot$ in Step 4.
- Bad_2: The event that $\mathcal{K}$ returns $\bot$ in Step 5(a)ii for some i, j.
- Bad_3: The event that $\mathcal{K}$ returns $\bot$ in Step 5b for some i.

Note that we have

$$\Pr\left[\mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho, z) = \bot\right] = \Pr[\mathsf{Bad}_1] + \Pr[\mathsf{Bad}_2] + \Pr[\mathsf{Bad}_3]. \tag{55}$$

Below (Lem. 22 to 24), we upper bound each term in the RHS of Eq. (55).

Lemma 22. $\Pr[\mathsf{Bad}_1] \leq 1 - 4\gamma + \eta$

Proof of Lem. 22. Eq. (54) implies

$$\Pr\left[\mathcal{M}_{\lambda,\gamma,z}(\rho) = 1\right] \geq 8\gamma.$$

By the definition of $\mathsf{ProjImp}(\mathcal{M}_{\lambda,\gamma,z})$ and an averaging argument, we have

$$\Pr\left[p \geq 4\gamma : p \leftarrow \mathsf{ProjImp}(\mathcal{M}_{\lambda,\gamma,z})(\rho)\right] \geq 4\gamma.$$

By Item 1 of Lem. 20, we have

$$\Pr\left[\widetilde{p} \geq 4\gamma - \varepsilon : \widetilde{p} \leftarrow \mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}(\rho)\right] \geq 4\gamma - \eta.$$

This completes the proof of Lem. 22.

□

Lemma 23. $\Pr[\mathsf{Bad}_2] \leq \gamma + \mathsf{negl}(\lambda)$.

Proof of Lem. 23. Since $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}$ is (ε,η)-almost projective, we have

$$\Pr[\widetilde{p}_{1,1} < \widetilde{p} - 2\varepsilon] \leq \eta.$$

Note that the loop done in Step 5a of $\mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho, z)$ is identical to $\mathsf{RepairExpt}^{\varepsilon,\eta}_{\lambda,\gamma,\zeta',z}(1^T)$ except for an additional check in Step 5(a)ii. Since $\mathsf{RepairExpt}^{\varepsilon,\eta}_{\lambda,\gamma,\zeta',z}(1^T)$ is $\big(2\varepsilon, 2(\eta+1/T) + 4\sqrt{\eta} + \sqrt{\zeta'} + \mathsf{negl}(\lambda)\big)$-almost projective by Corollary 5, it holds for each $(i,j) \neq (1,1)$ that

$$\Pr\left[\widetilde{p}_{i,j} < \widetilde{p}_{i,j-1} - 2\varepsilon\right] \leq 2(\eta + 1/T) + 4\sqrt{\eta} + \sqrt{\zeta'} + \mathsf{negl}(\lambda) \leq \gamma/(nT') + \mathsf{negl}(\lambda)$$

where we used $\eta \leq (\gamma/(18nT'))^2$, $T = \lceil 6nT'\gamma^{-1} \rceil$, and $\zeta' \leq (\gamma/(3nT'))^2$.

Noting that

$$\eta \leq \left(\gamma/(18nT')\right)^2 \leq \gamma/(nT'),$$

the union bound over all $(i,j) \in [n] \times [T']$ gives Lem. 23.

□

Lemma 24. $\Pr[\mathsf{Bad}_3] \leq \gamma$.

Proof of Lem. 24. For each i, j, let $\rho_{i,j}$ be the state just before applying $\mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta'^{-1}}, \cdot, z)$-$\mathsf{Repair}^{\varepsilon,\eta}(\widetilde{p}_{i,j})$ in Step 5(a)iii. Note that whenever Step 5(a)iii is invoked, either of Bad_1 or Bad_2 has not occurred by that point, which implies $\widetilde{p}_{i,j} \geq 4\gamma - (2nT'+1)\varepsilon \geq 2\gamma$ where we used $\varepsilon \leq 2\gamma/(2nT'+1)$. Since $\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}$ is (ε,η)-almost projective by Item 2 of Lem. 20,

$$\Pr\left[\widetilde{p}'_{i,j} \geq 2\gamma - \varepsilon : \widetilde{p}'_{i,j} \leftarrow \mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}(\rho_{i,j})\right] \geq 1 - \eta.$$

Since we have $\Delta^{\varepsilon}_{\mathsf{Shift}}(\mathsf{API}^{\varepsilon,\eta}_{\mathcal{M}_{\lambda,\gamma,z}}, \mathsf{ProjImp}(\mathcal{M}_{\lambda,\gamma,z})) \leq \eta$ by Item 1 of Lem. 20, we have

$$\Pr\left[p_{i,j} \geq 2\gamma - 2\varepsilon : p_{i,j} \leftarrow \mathsf{ProjImp}(\mathcal{M}_{\lambda,\gamma,z})(\rho_{i,j})\right] \geq 1 - 2\eta.$$

This implies

$$\Pr\left[d = \top \ : \ d \leftarrow \mathcal{V}(1^\lambda, 1^{\gamma^{-1}}, \rho_{i,j}, z)\right] \geq (1-2\eta)(2\gamma - 2\varepsilon) \geq \gamma$$

where we used $\eta \leq (\gamma/(18nT'))^2 \leq \gamma/4$ and $\varepsilon \leq \gamma/4$. Thus, by **Assumption 2** of Lem. 18, we have

$$\Pr\left[s_i = s^*_{z,i} \ : \ s_i \leftarrow \mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta'^{-1}}, \rho_{i,j}, z)\right] \geq \delta - \zeta' - \mathsf{negl}(\lambda) \geq \delta/3$$

for sufficiently large λ where we used $\zeta' \leq \delta/2$. Noting that $\mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta'^{-1}}, \cdot, z)$-$\mathsf{Repair}^{\varepsilon,\eta}(1^T, \widetilde{p}_{i,j})$ on $\rho_{i,j}$ yields the identical distribution as $\mathcal{K}_i(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta'^{-1}}, \rho_{i,j}, z)$ by Corollary 5 , for each i, j, the probability of breaking the inner loop in Step 5(a)iv is at least $\delta/3$. Thus, for each i, the probability that this does not happen for all $j \in [T']$ is at most $(1-\delta/3)^{T'} \leq \gamma/n$. By taking the union bound over all $i \in [n]$, Lem. 24 holds.

□

Combining Eq. (55) and Lem. 22 to 24, we have

$$\Pr\left[\mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho, z) = \bot\right] \leq 1 - 4\gamma + \eta + \gamma + \gamma + \mathsf{negl}(\lambda) \leq 1 - \gamma + \mathsf{negl}(\lambda)$$

where we used $\eta \leq (\gamma/(18nT'))^2 \leq \gamma$. Combined with Property 1 of Lem. 18 which is already proven, we have

$$\Pr\left[\mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \rho, z) = s^*_{z,1}||...||s^*_{z,n}\right] \geq \gamma - \zeta - \mathsf{negl}(\lambda).$$

This finishes the proof that $\mathcal{K}$ satisfies Property 2 of Lem. 18.

On Strictly QPT. Finally, we argue how to convert $\mathcal{K}$ into a *strict* QPT one. For some polynomial $C(\lambda)$, suppose that we modify $\mathcal{K}$ so that if it runs $C(\lambda)$ times longer than its expected run time, then it immediately outputs $\bot$ and halts. Then $\mathcal{K}$ now runs in strict QPT. This modification does not affect Property 1 since $\mathcal{K}$ only outputs $\bot$ in the case of the time out. By Markov's inequality, the time out occurs with probability at most $C(\lambda)^{-1}$, which may decrease the probability in Property 2 by at most $C(\lambda)^{-1}$. Thus, if we set $C(\lambda)$ in such a way that $C(\lambda)^{-1} \leq \delta'(\lambda)/2$, then Property 2 is still satisfied if we replace $\delta'(\lambda)$ with $\delta'(\lambda)/2$.

This completes the proof of Lem. 18.

10 Post-Quantum Non-Malleable Commitments: One-Many

In this section, we turn to the construction of a black-box commitment scheme that satisfies the weak *one-many* definition of post-quantum non-malleability (as described in Def. 6). The construction for this setting is identical to Prot. 3 (more accurately, its 'two-sided' version described in Sec. 8 and Prot. 10) given in Sec. 5.1, and thus uses the same component primitives. In effect, we show that the security of Prot. 3 extends to the one-many case as well. This is stated formally below.

Theorem 11. *For any polynomial $N(\lambda)$ in the security parameter, Prot. 10 is a black-box, constant-round construction of a 1-N post-quantum weakly non-malleable commitment (as per Def. 6) in the synchronous setting, supporting tag space $[T]$ with $T(\lambda)$ being any polynomial in λ.*

Note that in Thm. 11, we have referred to the number of right sessions in the non-malleability game by N, while this parameter was denoted by k in Prot. 3. We rename this for clarity, since k is already used to denote an important quantity in our construction.

To prove Thm. 11, note that completeness and hiding can be argued similar as for Prot. 3. We focus on showing 1-many non-malleability in the following.

10.1 Proof Overview

Technically, we need to prove weak 1-many non-malleability for Prot. 10. But to simplify the presentation, we only focus on the 'one-sided' version of it (i.e., Prot. 3) in the following. This is because we can use the same 'two-slots' trick as explained in Sec. 8 to lift the proof to the 'two-sided' setting as well.

While the setting is now different for the 1-many case, and our formalism for handling this case has to change to account for this, we wish to emphasize that our *core strategy* and indeed our intuition for this proof remains essentially the same! Intuitively, the fundamental idea behind our proof is to design experiments that allow us to reduce non-malleability to the *hiding* property of the initial commitment on the left. Now while the 1-many setting involves multiple parallel right

sessions, there is only a single session on the left, and so our proof strategy and technique remains largely unchanged—At a high level, we seek to extract a 'trapdoor' from Step 2c of the left session to then decouple the main body of the left session from the commitment made in the prefix phase (i.e., Step 1), and subsequently we extract the value from the Step 3 ExtCom of the right session and show that this must be the committed value $\widetilde{m}$ in the right session with sufficiently high probability.

This helps us better understand what remains unchanged (in the 1-many case) from our earlier proof for Prot. 3, and what needs new treatment. The key thing that changes here is the extraction step, which is now more involved. Note that to establish 1-many non-malleability, we must obtain the tuple of values committed by $\mathcal{M}$ *across* all the N right sessions (and show this must not depend on the left session commitment). So in our proof, we must extract the right side committed value as before, but now we must do so *simultaneously* from all the parallel sessions with sufficiently high probability. It is not clear that our technique so far extends directly to this case; indeed, this extension to the *simultaeneous extraction* case turns out to be nontrivial and will be our focus here.

More technically, we will arrange our task as follows:

- First, we describe how to capture the reduction from non-malleability to the VSS hiding game Expr. 1 by modifying the MIM experiment. This is essentially identical to what is presented in Sec. 5.3 and 5.4. We will sketch the argument while mostly focusing on the syntactic differences arising from the 1-many setting.
- We state the key lemma formalizing the existence of the simulator-extractor (i.e., the analog of $\mathcal{SE}$ from Sec. 5.4) in the 1-many setting. This is used to complete the proof of non-malleability.
- We then show how to build an *instanced* version of the simulation-less extractor $\mathcal{K}$ from Lem. 11: *For each session $j \in [N]$ on the right*, we design an extractor $\mathcal{K}^{(j)}$ that works essentially the same way as $\mathcal{K}$ in the 1-1 setting (i.e., in Lem. 11).
- Finally, we show how to get a full-fledged simulation-extractor from these $\mathcal{K}^{(j)}$'s using our simultaneous extraction lemma given in Lem. 18.

10.2 Reduction to VSS Hiding Game

We start by describing the analogs of the key experiments introduced in Sec. 5.3. We begin by defining the real man-in-the-middle experiment.

Game $H^{\mathcal{M}_\lambda}(\lambda, m, N, \rho_\lambda)$: Analogous to Algo. 5.1, this is just the real MIM interaction, now in the 1-N setting. The game now takes the number of parallel right sessions N as input. We continue to denote the left committer by C. But for clarity, we now refer to the various receivers on the right by $R^{(1)}, \ldots, R^{(N)}$ respectively. Recall that in the 1-N man-in-the-middle experiment, these receivers function honestly and independently.

The output of this game is again denoted by $\mathsf{Out}_{H^{\mathcal{M}_\lambda}}(\lambda, m, \rho_\lambda)$ and consists of the following parts:

1. OUT: The (quantum) output of $\mathcal{M}$ at the end of this game;
2. For each $j \in [N]$, $\widetilde{\tau}^{(j)}$: The commitment transcript sent by $\mathcal{M}$ to $R^{(j)}$ in the Step 1 of the j-th right session;
3. Also for each $j \in [N]$, $\widetilde{d}^{(j)} \in \{\top, \bot\}$: The output of the honest receiver $R^{(j)}$ in the j-th right session, indicating if $\mathcal{M}_\lambda$'s commitment in the j-th right session is accepted ($\widetilde{d}^{(j)} = \top$) or not ($\widetilde{d}^{(j)} = \bot$).

We will also need to refer to the value committed in the right sessions. Toward that, we define the function $\Gamma_{\{\widetilde{d}^{(j)}\}_{j=1}^{k}}(\cdot)$ in exactly the style given in Def. 6.

Specifically, let $\mathsf{val}^{(j)}(\widetilde{\tau}) = \mathsf{val}^{(j)}(\widetilde{\tau}^{(j)})$ denote the value committed in the j-th right session by $\mathcal{M}$ (note that this is well defined by the statistical binding property).

Thus, to prove 1-N non-malleability as per Def. 6, we need only establish the following:

$$\begin{aligned}&\left\{\Gamma_{\{\widetilde{d}_0^{(j)}\}_{j=1}^{N}}\left(\mathsf{OUT}_0, \{\mathsf{val}^{(j)}(\widetilde{\tau}_0^{(j)})\}_{j\in[N]}\right) : (\mathsf{OUT}_0, \{\widetilde{\tau}_0^{(j)}, \widetilde{d}_0^{(j)}\}_{j\in[N]}) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m_0, N, \rho_\lambda)\right\}\\ &\overset{c}{\approx} \left\{\Gamma_{\{\widetilde{d}_1^{(j)}\}_{j=1}^{N}}\left(\mathsf{OUT}_1, \{\mathsf{val}^{(j)}(\widetilde{\tau}_1^{(j)})\}_{j\in[N]}\right) : (\mathsf{OUT}_1, \{\widetilde{\tau}_1^{(j)}, \widetilde{d}_1^{(j)}\}_{j\in[N]}) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m_1, N, \rho_\lambda)\right\}, \qquad (56)\end{aligned}$$

where both ensembles are indexed by $\lambda \in \mathbb{N}$ and $(m_0, m_1) \in \{0,1\}^{\ell(\lambda)} \times \{0,1\}^{\ell(\lambda)}$.

We next turn to defining the machine $\widetilde{H}^{\mathcal{M}_\lambda}$ in the 1-N setting, which is the analog of Algo. 5.2 in Sec. 5.3.

Algorithm 10.1: Game $\widetilde{H}^{\mathcal{M}_\lambda}(\lambda, \varepsilon, m, N, \rho_\lambda)$ in 1-N Setting

Input: This takes as input the same parameters λ, ρ_λ, m, and N as for $H^{\mathcal{M}_\lambda}$. It additionally takes as input a noticeable function $\varepsilon(\cdot)$.
It proceeds as follows:

1. **(Prefix phase.)** This proceeds as follows.
   (a) Sample a random size-k subset $\eta \subset [n]$.
   (b) Execute $H^{\mathcal{M}_\lambda}(m, N, \lambda, \rho_\lambda)$ until the end of Step 1. At the moment, it already receives the Step 1 commitment made by the left-session honest committer C. It performs brute-force computation to obtain from C's commitment the committed shares cv_i and their decommitment information for $i \in \eta$. We denote these values as $\mathsf{VI}_\eta := \{(\mathsf{cv}_i, \mathsf{decom}_i\}_{i\in\eta}$.

   Notation: Let $\mathsf{st}_{\mathcal{M}}$ denote the state of $\mathcal{M}$ at the end of Step 1; Let st_C (and $\{\mathsf{st}_R^{(j)}\}_{j\in[N]}$) denote the state of the honest committer (and receivers) at the end of Step 1; Let $\{\widetilde{\tau}^{(j)}\}_{j\in[N]}$ denote the commitments sent by $\mathcal{M}$ in Step 1 in all the N right sessions. We denote the tuple $(\mathsf{st}_{\mathcal{M}}, \{\mathsf{st}_R^{(j)}\}_{j\in[N]}, \tau, \{\widetilde{\tau}^{(j)}\}_{j\in[N]})$ as $\mathsf{pref}^{1:N}$. We use the following nation to express the execution of this **Prefix phase**:

$$\mathsf{pref}'^{1:N} := (\mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pref}}(\lambda, m, N, \rho_\lambda). \qquad (57)$$

   We remark that this prefix generation step is independent of the error parameter ε.
2. **(Remainder phase.)** This involves the following steps:
   (a) $\widetilde{H}^{\mathcal{M}_\lambda}$ invokes the $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta))$ defined in Algo. 10.2 (where $\mathsf{pref}^{1:N}$ is now as defined above in the 1-N setting), which now outputs a tuple $(\mathsf{OUT}, \{\widetilde{d}^{(j)}\}_{j\in[N]})$.
   (b) $\widetilde{H}^{\mathcal{M}_\lambda}$ outputs $(\mathsf{OUT}, \{\widetilde{\tau}^{(j)}\}_{j\in[N]}, \{\widetilde{d}^{(j)}\}_{j\in[N]})$.

We now describe the subprocedure $\mathcal{G}_1(\cdot)$ adapted to the 1-N setting, which is the analog of Algo. 5.3 in Sec. 5.3.

Algorithm 10.2: Machine $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)$ in 1-N Setting

Machine $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)$ works in the same manner as the $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)$ defined in Algo. 5.3 but in the 1-N setting. This is even no need to give a full description of the current

$\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)$, because it has literally identical *syntax* as the $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)$ defined in Algo. 5.3—All the $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}, \eta, \mathsf{VI}_\eta)$ does is to make some modifications on the left session; Here in the 1-N setting, we also has only a single left session. So, what our current $\mathcal{G}_1(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)$ does is to do the same thing on the left session as Algo. 5.3 and follow the honest receivers' algorithm on the right sessions.
The only point that deserves a remark is the format of the output of the current $\mathcal{G}_1$ in the 1-N setting, which we describe as follows:

– It finally outputs the values $(\mathsf{OUT}, \{\widetilde{d}^{(j)}\}_{j\in[N]})$, where again OUT is $\mathcal{M}$'s final output and $\{\widetilde{d}^{(j)}\}_{j\in[N]}$ are the final decisions by the honest $R^{(j)}$s in the N right session.

As in Sec. 5.3, one can again show that the outputs of H and $\widetilde{H}$ defined above are computationally indistinguishable (i.e., an analog of Lem. 8). We next turn our attention to the machine $\widetilde{G}$ in the 1-N setting that makes use of our *simultaeneous* simulator-extractor. This machine is the analog of Algo. 5.4 in Sec. 5.3.

Algorithm 10.3: Game $\widetilde{G}^{\mathcal{M}_\lambda}(\lambda, m, N, \rho_\lambda, \varepsilon)$

This proceeds in two phases as well:

1. **(Prefix phase.)** This is identical to the prefix phase of Algo. 5.2, i.e., it computes $(\mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta) \leftarrow \widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pref}}(\lambda, m, N, \rho_\lambda)$.
2. **Remainder phase:** This involves the following steps:
   – It invokes a machine $\mathcal{SE}$, which is guaranteed to exist by the following Lem. 25: $\mathcal{SE}$ takes in as input a tuple $(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)$ and outputs $(\mathsf{OUT}, \{\mathsf{Val}^{(j)}\}_{j\in[N]})$.
   – $\widetilde{G}^{\mathcal{M}_\lambda}$ outputs $(\mathsf{OUT}, \{\mathsf{Val}^{(j)}\}_{j\in[N]})$ as its own output.

With this, we can turn to the main lemma: comparing executions of $\widetilde{H}$ and $\widetilde{G}$. Note that in the latter game we also obtain the committed values in the right sessions. Then Lem. 25 that is analogous to Lem. 9 shows that these executions yield outputs that are close up to a controllable error parameter:

Lemma 25 (1-many Simulation-Extractor). *Let $\mathcal{G}_1(\cdot)$ be the efficient procedure defined in Algo. 10.2. For any polynomial $N(\lambda)$ in the security parameter λ, there exists a simulation-extractor $\mathcal{SE}$ such that for any $(\mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)$ in the support of $\widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pref}}$, and for any noticeable $\varepsilon(\lambda)$, there is a noticeable $\varepsilon'(\lambda) \le 8\varepsilon(\lambda)$ that is efficiently computable from $\varepsilon(\lambda)$ such that the following holds:*

$$\begin{aligned}&\left\{(\mathsf{OUT}, \{\mathsf{Val}^{(j)}\}_{j\in[N]}) : (\mathsf{OUT}, \{\mathsf{Val}^{(j)}\}_{j\in[N]}) \leftarrow \mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)\right\}\\ \overset{c}{\approx}_\varepsilon &\left\{\Gamma_{\{\widetilde{d}^{(j)}\}_{j=1}^N}\left(\mathsf{OUT}, \{\mathsf{val}^{(j)}_{\widetilde{d}}(\widetilde{\tau})\}_{j\in[N]}\right) : (\mathsf{OUT}, \{\widetilde{d}^{(j)}\}_{j\in[N]}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon'^{-1}}, \mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)\right\}.\end{aligned}$$

The remainder of the proof to non-malleability can be shown just as in the earlier setting (via the reduction to the VSS hiding game as done in Sec. 5.4). We omit the details to avoid repetition. Instead, we will focus on the key task of *building this simulator-extractor $\mathcal{SE}$*. Our approach will be the same as before—Namely, for each right session $j \in [N]$, we will first describe a base extractor $\mathcal{K}^{(j)}$ that is essentially the extractor $\mathcal{K}$ from Lem. 11 but localized to session j, and then show (in Sec. 10.4) how to use these 'localized' versions of $\mathcal{K}$ to obtain the 1-N simulation-extractor $\mathcal{SE}$ as required by Lem. 25.

10.3 Localized Simulation-Less Extractors $\mathcal{K}^{(j)}$

Recall that we defined the shorthand $\mathsf{pref}'^{1:N} := (\mathsf{pref}^{1:N}, \eta, \mathsf{VI}_\eta)$. Also, we define the following quantity $p^{\mathsf{Sim}}_{\mathsf{pref}'^{1:N}}[\varepsilon_1]$ that will be important in the formal statement of $\mathcal{K}^{(j)}$s. It is the 1-N analog of the quantity $p^{\mathsf{Sim}}_{\mathsf{pref}'}[\varepsilon_1]$ defined in Eq. (11).

$$p^{\mathsf{Sim}}_{\mathsf{pref}'^{1:N}}[\varepsilon_1] := \Pr\Big[\wedge_{j\in[N]}\big(\widetilde{d}^{(j)} = \top\big) : (\mathsf{OUT}, \{\widetilde{d}^{(j)}\}_{j\in[N]}) \leftarrow \mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}'^{1:N})\Big]. \tag{58}$$

The following lemma states the formal guarantee provided by these localized base extractors $\mathcal{K}^{(j)}$. It is the 1-N analog of lem. 11.

Lemma 26 (Localized Simulation-less Extraction). *For any polynomial $N(\lambda)$, let $\widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pref}}(\lambda, m, N, \rho_\lambda)$ be as defined in Algo. 10.1. There exist QPT machines $\{\mathcal{K}^{(j)}\}_{j\in[N]}$ such that for any noticeable $\varepsilon(\lambda)$, there is a noticeable $\varepsilon_1(\lambda) \le \varepsilon(\lambda)$ that can be efficiently computed from ε, such that for any noticeable $\varepsilon_2(\lambda)$ and any tuple $\mathsf{pref}'^{1:N} = (\mathsf{st}_{\mathcal{M}}, \{\mathsf{st}_{R^{(j)}}\}_{j\in[N]}, \tau, \{\widetilde{\tau}^{(j)}\}_{j\in[N]}, \eta, \mathsf{VI}_\eta)$ in the support of $\widetilde{H}^{\mathcal{M}_\lambda}_{\mathsf{pref}}(\lambda, m, N, \rho_\lambda)$, the following conditions hold:*

1. **(Almost Uniqueness:)** *For each $j \in [N]$, $\mathcal{K}^{(j)}$ takes as input $(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}'^{1:N})$. It outputs a value $\mathsf{Val}^{(j)} \in \{0,1\}^{\ell(\lambda)} \cup \{\bot\}$ that satisfies*

$$\Pr\Big[\mathsf{Val}^{(j)} \notin \{\mathsf{val}(\widetilde{\tau}^{(j)}), \bot\} \ : \ \mathsf{Val}^{(j)} \to \mathcal{K}^{(j)}(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}'^{1:N})\Big] \le \varepsilon_2(\lambda) + \mathsf{negl}(\lambda).$$

2. **(Extraction:)** *If $p^{\mathsf{Sim}}_{\mathsf{pref}'^{1:N}}[\varepsilon_1] \ge \varepsilon(\lambda)$, then for each $j \in [N]$ it holds that*

$$\Pr\Big[\mathsf{Val}^{(j)} = \mathsf{val}(\widetilde{\tau}^{(j)}) : \mathsf{Val}^{(j)} \leftarrow \mathcal{K}^{(j)}(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}'^{1:N})\Big] \ge \frac{\varepsilon'(\lambda) - \varepsilon_2(\lambda)}{\widetilde{t}},$$

where $p^{\mathsf{Sim}}_{\mathsf{pref}'^{1:N}}[\varepsilon_1]$ is defined in Eq. (58) and $\varepsilon'(\lambda) := \frac{\varepsilon(\lambda)}{10t^2}$.

This statement of the simulation-less extractor(s) in the one-many setting looks quite different from the one appearing in Lem. 11, and at first glance, seems to be significant extension of the latter. At the very least, there are more moving parts with the multiple parallel sessions on the right. Further consideration however reveals that this is not quite the case. The trick to this is to see that since the receivers $R^{(1)}, \ldots R^{(N)}$ in the right sessions operate *independently*, and act honestly (unless we extract in that session), we can simply treat them as standard interactions of the commitment.

In particular, when invoking this extractor on a particular session $j \in [N]$, we can treat the other right sessions as *context*: in more detail, given the 1-many man-in-the-middle adversary $\mathcal{M}$, we can come up with a new adversary $\mathcal{M}^{(j)}$ that runs the other parallel sessions internally in a one-many MIM interaction with $\mathcal{M}$ and then treats the j-th session as the *sole* right session in a one-one MIM interaction. We can then apply Lem. 11 to derive these guarantees for $K^{(j)}$ (where the MIM adversary is $\mathcal{M}^{(j)}$). Armed with this reasoning, we can see that Lem. 26 readily follows from Lem. 11.

This also allows us to set parameters the same way as in Lem. 11. Namely, we work by first fixing a noticeable $\varepsilon(\cdot)$ and then set $\varepsilon_1(\lambda) := \frac{t+1}{t^2+4t+2} \cdot \varepsilon'(\lambda)$ with $\varepsilon'(\lambda) := \frac{\varepsilon(\lambda)}{10t^2}$.

10.4 Simulation-Extractor $\mathcal{SE}$: 1-Many Setting

To finish the proof of Lem. 25 (and thus the proof of weak 1-many non-malleability), there are two things left now. First, we will build a *simultaneous* simulation-less extractor $\mathcal{K}$ that extracts the $\widetilde{m}^{(j)}$'s in all the right sessions. As long as we have such an $\mathcal{K}$, we can re-use the noisy simulation extraction lemma (i.e., Lem. 17) to upgrade it to the desired simulation-extractor $\mathcal{SE}$ as required by Lem. 25. In the following, we elaborate on these two steps.

Simultaneous Simulation-less Extractor $\mathcal{K}$. Such a $\mathcal{K}$ can be built by applying the simultaneous extraction lemma we developed in Lem. 18 to the localized simulation-less extractors $\mathcal{K}^{(j)}$'s. For that, we first need to prove that these $\mathcal{K}^{(j)}$'s in Lem. 26 indeed satisfied the prerequisites in Lem. 18. Similar as in Sec. 7.2, we will not show it directly with the $\mathcal{K}^{(j)}$'s. Some 'wrapper' machines need to be defined to make the parameters match. Fortunately, this step is almost identical to what we did in Sec. 7.2. We will be able to use almost the same parameter settings.

Machine $\mathcal{G}'$: it takes as input $(1^\lambda, 1^{\gamma^{-1}}, \mathsf{pref}'^{1:N})$ and proceeds as follows:

1. Set $\varepsilon := \gamma$.
2. Compute ε_1 from ε. Note that this can be done because Lem. 26 stipulates the there is a noticeable $\varepsilon_1 \leq \varepsilon$ that is efficiently computable from ε.
3. Run machine $\mathcal{G}_1(1^\lambda, 1^{\varepsilon_1^{-1}}, \mathsf{pref}'^{1:N})$ (as per Lem. 26) and output whatever it outputs.

Machine $\mathcal{K}'^{(j)}$ $(j \in [\widetilde{t}])$: it takes as input $(1^\lambda, 1^{\gamma^{-1}}, 1^{\zeta^{-1}}, \mathsf{pref}'^{1:N})$ and proceeds as follows:

1. Set $\varepsilon := \gamma$.
2. Compute ε_1 from ε. Note that this can be done because Lem. 26 stipulates the there is a noticeable $\varepsilon_1 \leq \varepsilon$ that is efficiently computable from ε.
3. Set $\varepsilon_2 := \zeta$.
4. Run machine $\mathcal{K}^{(j)}(1^\lambda, 1^{\varepsilon_1^{-1}}, 1^{\varepsilon_2^{-1}}, \mathsf{pref}'^{1:N})$ (as per Lem. 26) and output whatever it outputs.

Compare the above machines with the $\mathcal{G}'$ and $\mathcal{K}'$ defined in Sec. 7.2, the only differences regarding the parameters is that we set $\varepsilon_1 = \gamma$ directly, instead of $\varepsilon_1 = 8\gamma$. That is because the RHS of the **Assumption 2** of Lem. 18 is $\gamma(\lambda)$ (instead of $8\gamma(\lambda)$ in the RHS of Item 2 in Lem. 17).

Now, if we treat the above $\mathcal{G}'$ as machine $\mathcal{V}$ in Lem. 18, treat the above $\{\mathcal{K}'^{(j)}\}_{j\in[N]}$ as the machines $\{\mathcal{K}_i\}_{i\in[n]}$ (i.e., $n = N$), and set $\delta(\lambda) := \frac{\varepsilon'(\lambda)}{\widetilde{t}}$, then it is straightforward to see that that **Assumption 1** and **Assumption 2** of Lem. 18 are satisfied.[36] Thus, Lem. 18 implies the desired *simultaneous* simulation-less extractor $\mathcal{K}$ that is able to extract all the committed values in the N right sessions.

1-Many Simulation-Extractor $\mathcal{SE}$. Finally, simply observe that Properties 1 and 2 in Lem. 18 is exactly the prerequisites of the noisy simulation-extraction lemma (i.e., Lem. 17). Thus, a straightforward application of Lem. 17 to the machines $\mathcal{K}$ and $\mathcal{V}$ (with them being the $\mathcal{K}$ and $\mathcal{G}$ in Lem. 17) implies our desired simulation-extraction $\mathcal{SE}$ in the 1-N setting. This finishes the proof of Lem. 25 , and thus in turns finishes our proof of weak 1-N non-malleability (i.e., Def. 6).

Note that limitation to weak 1-many non-malleability is somewhat inherent to our approach: the simultaeneous simulator-extractor we define has a requirement on the machine that corresponds to $\mathcal{G}'$ — namely, it should output $\top$ with a certain noticeable probability. In our interpretation,

[36] Similar as in Sec. 7.2, $\mathsf{pref}'^{1:N}$, $\{\widetilde{\tau}^{(j)}\}_{j\in[N]}$, and $\{\mathsf{val}(\widetilde{\tau}^{(j)})\}_{j\in[N]}$ play the role of, ρ_λ, z_λ, and $\{s^*_{z_\lambda,j}\}_{j\in[N]}$ respectively.

this output corresponds to the *conjunction* of all the verifier decisions (accept/reject) in the right sessions. This condition translates to assuming that our MIM adversary completes every right session successfully with a reasonably large probability, and so our treatment can only consider adversaries that obey this constraint — and not ones, for example, that *always* abort in certain right sessions. We thus eschew showing standard 1-many non-malleability; and as we shall later see, weak one-many non-malleability suffices for the applications we have in mind for our commitment.

11 Post-Quantum Multi-Party Parallel OT

In this section, we describe a black-box, constant-round protocol implementing the n-party OT functionality (as defined in Fig. 1) w.r.t. the ε-simulatable PQ-MPC security notion (as per Def. 7). We state the definition we will be able to realize and work-with in Def. 18.

Figure 1: The Ideal Functionality $\mathcal{F}_{\mathbf{OT}}^{n}$

The functionality $\mathcal{F}_{\mathrm{OT}}^{n}$ is specified by the number of distinct senders S_i and receivers R_j (for $i, j \in [n]$). It acts as follows:

Sender's Message: $\mathcal{F}_{\mathrm{OT}}^{n}$ receives $(\mathsf{send}, i, j, s_0^i, s_1^i)$ from a sender S_i (which also specifies a purported receiver R_j). It ignores this message if $i = j$. Otherwise, it records this quintuple, and ignores subsequent messages that have the same initial triple (send, i, j).

Receiver's Message: $\mathcal{F}_{\mathrm{OT}}^{n}$ receives $(\mathsf{receive}, i, j, b)$ from a receiver R_j. It responds with $(\mathsf{open}, i, j, s_b^i)$ to the receiver R_j.

Definition 18 (Post-Quantum Multi-Party OT with ε-Simulation). *A protocol Π is called a malicious multiparty secure oblivious transfer protocol if for every polynomial $n := n(\lambda)$, Π is a post-quantum (ε-simulatable) MPC protocol for $\mathcal{F}_{OT}^{n}$ precisely as stated in Def. 7.*

Below we will describe a constant-round, black-box protocol and prove that it satisfies Def. 18. Our approach will in fact be to first describe a *2-party* OT protocol Π running in *constant rounds*, and then show that the *parallel repetition* of Π satisfies the requirements of Def. 18.

More precisely for a 2-party protocol Π, let us define the *n-fold parallel repetition* as follows: consider n entities where each entity would like to participate in an OT session *both* as a sender and as a receiver against all the other entities in parallel. In other words, for each $i \neq j \in [n]$, we consider two OT sessions between parties P_i and P_j where in one P_i plays the role of sender and P_j that of the receiver, and vice versa in the other. This comprises $2 \cdot \binom{n}{2}$ parallel independent executions of Π. We will show that the n-fold parallel repetition of Π (denoted by Π^n) is a post-quantum (ε-simualatable) MPC protocol for $\mathcal{F}_{\mathrm{OT}}^{n}$.

We note that sequential composition would serve the same purpose were we not constrained by the constant-round requirement. Relying on parallel composition, however, we get the desired constant-round OT protocol since Π^n has the same round complexity as Π.

11.1 Building Blocks

Before diving into the main construction of the parallel malicious secure OT protocol, we need some building blocks.

Malicious-Sender Secure OT: The first component we require is a constant-round OT protocol with a simpler or weaker property: namely, with security against *malicious senders* and *semi-honest*

receivers. Additionally, we require that the associated simulator for proving security be straightline. Fortunately, such schemes are known to be easily obtainable from any of the following: post-quantum dense cryptosystems, post-quantum linearly homomorphic PKE, or post-quantum lossy PKE (see, e.g., [CDMW09, Wee10]). These schemes are black-box constructions and also work in the post-quantum setting (due to straightline security proofs).

Theorem 12 ([CDMW09, Wee10]). *There exist two-round and black-box post-quantum OT schemes with indistinguishability security against honest receivers and simulation security against malicious QPT senders, based on (any of) post-quantum dense cryptosystems, linearly homomorphic public key encryption, or lossy public key encryption.*

We will denote such a protocol by Γ, where the sender uses as inputs two strings (s_0, s_1) and receiver uses as input a bit r. For technical reasons we will also refer in our construction to the receiver's private random tape in the protocol, which we denote as a string τ of length $t(\lambda)$ that is a polynomial in the security parameter.

Post-Quantum Extractable Commitment: We make use of the post-quantum parallelly extractable commitment scheme ExtCom with ε-simulation (as per Def. 4), which can be built in black-box from any post-quantum OWFs (see Lem. 3).

1-Many Weak Non-Malleable Commitment: The final component we require is a constant-round, post-quantum, *1-many weakly* non-malleable commitment scheme that is also parallel ε-simulation extractable. To make our overall OT construction fully black-box, we also require this construction to be fully black-box. Fortunately, such a construction is available to us from Sec. 10.

We note that the extractablity property mentioned above is easily observed due to the intrinsic execution of ExtCom in Step 3 of Prot. 3, for which we can invoke the associated $\mathcal{SE}_{\mathsf{ExtCom}}$ (it is easy to see that this step also commmits to the initial committed value with overwhelming probability). Indeed, such an observation was made in [LPY23a]. We denote this protocol by ENMC.

11.2 Construction

Our construction is given below in Prot. 4.

Protocol 4: The parallel malicious secure OT scheme Π

Parameters: The security parameter is denoted by λ. Other parameters will be specified by polynomials in λ unless otherwise specified.
Receiver's input: A bit $r \in \{0,1\}$
Sender's input: Strings $s_0, s_1 \leftarrow \{0,1\}^\ell$
The protocol proceeds as follows:

1. **Phase I: Random Tape Coin Tossing**
   (a) The receiver samples 2λ uniform random strings $(r_1^R, \tau_1^R), \ldots, (r_{2\lambda}^R, \tau_{2\lambda}^R)$ of length $t+1$ corresponding to samples of the receiver's input bit and randomness for Γ.
   (b) The receiver then runs 2λ parallel executions of ExtCom with the sender, where the receiver commits to the values $(r_1^R, \tau_1^R), \ldots, (r_{2\lambda}^R, \tau_{2\lambda}^R)$ independently.
   (c) The sender then samples 2λ uniform random strings $(r_1^S, \tau_1^S), \ldots, (r_{2\lambda}^S, \tau_{2\lambda}^S)$ of its own and sends these back to the receiver.
   (d) The receiver sets $r_i = r_i^R \oplus r_i^S$ and $\tau_i = \tau_i^R \oplus \tau_i^S$ for $i \in [2\lambda]$.
2. **Phase II: Base OT Execution**

(a) The sender samples 2λ pairs of uniform random strings $(s_1^0, s_1^1), \ldots, (s_{2\lambda}^0, s_{2\lambda}^1)$.

(b) The sender and receiver then execute 2λ parallel executions of Γ. For the ith execution, the sender uses the inputs (s_i^0, s_i^1) and the receiver uses input r_i and randomness τ_i (for each $i \in [2\lambda]$).

3. **Phase III: Cut & Choose**

(a) The sender samples an uniform random string $q_S \xleftarrow{\$} \{0,1\}^\lambda$. It then runs an execution of ENMC with the receiver to commit to the string q_S.

(b) The receiver then samples an uniform string $q_R \xleftarrow{\$} \{0,1\}^\lambda$ and sends this back to the sender.

(c) The sender provides the *decommitment* of its commitment to q_S to the receiver.

(d) Both parties then compute $q = q_S \oplus q_R$. They also compute the description of a subset $Q \subset [2\lambda]$ of size λ using the following correspondence: $Q = \{2i - q_i\}_{i=1}^{\lambda}$ where q_i is the ith bit of q. More descriptively, we imagine the previous 2λ executions of Γ to be laid out in λ pairs (of adjacent executions). Then Q marks the subset of executions to be 'opened', including the first or second execution in each of the λ pairs depending on whether q_i is 0 or 1 (so Q has exactly one member in each pair).

(e) For each $i \in Q$, the receiver *decommits* its phase I commitment to (r_i^R, τ_i^R).

(f) The sender then computes (r_i, τ_i) for all such sessions $i \in Q$. It next checks that (r_i, τ_i) is consistent with the receiver's messages in the ith parallel session of Γ in phase II. The sender aborts if this is not the case.

4. **Phase IV: OT Combiner**

(a) For every $j \notin Q$, the receiver computes $\alpha_j = r \oplus r_j$ (recall r is its original input bit) and sends the list $\{\alpha_j\}_{j \notin Q}$ to the sender.

(b) The sender then computes $\sigma_0 = s_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and $\sigma_1 = s_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$. It sends (σ_0, σ_1) to the receiver.

(c) Finally, the receiver computes and outputs the string $s_r = \sigma_r \oplus (\bigoplus_{j \notin Q} s_j^{r_j})$.

Remark 7. We note here that we can use an ENMC scheme as described above in lieu of ExtCom in Step 1 without losing anything in terms of functionality (indeed, earlier works do exactly this). We use the extractable commitment ExtCom separately for modularity and to invite a clearer examination of how the separate components and assumptions are used in our construction and what role they play in security.

11.3 Security

The correctness of Prot. 4 is straightforward. We turn to proving security for this scheme. This is captured formally by the following theorem.

Theorem 13. *Let λ denote the security parameter. Let Γ, ExtCom, and ENMC be as described in Sec. 11.1. Then, the n-fold parallel execution of Prot. 4 (i.e., an execution among n parties, where each pair of parties run two independent parallel instances with reversed roles of sender and receiver) realizes Def. 18, for any polynomial $n = n(\lambda)$.*

Algorithm 11.1: The simulator $\mathcal{S}$ for the parallel malicious secure OT protocol

Description: The simulator $\mathcal{S}$ enjoys black-box access to the possibly quantum adversary $\mathcal{A}$. The simulator may interact using quantum communication with the adversary occasionally (indeed, this is required to carry out the quantum analog of rewinding). It also takes in as input the security parameter as 1^λ, and the error parameter as $1^{1/\varepsilon}$. Thus its runtime is $\mathsf{poly}(\lambda, \varepsilon^{-1})$.

Operation: Note that the simulation is for an n-fold parallel execution of Π, where $\mathcal{A}$ may corrupt a single party in each session of Π. In each session, $\mathcal{S}$ will act according to which of the parties $\mathcal{A}$ corrupts. Its operation is detailed below.

Sender Corruption: $\mathcal{S}$ takes over the receiver operation for the session. It acts as follows:

- **Preamble:** It starts by sampling an uniform string $q \xleftarrow{\$} \{0,1\}^\lambda$ and computing the associated subset $Q \subset [2\lambda]$.
- **Phase I:** For each $i \in Q$, $\mathcal{S}$ acts just as the honest receiver for this phase, committing to uniformly sampled (r_i^R, τ_i^R). However, for each $i \notin Q$, $\mathcal{S}$ commits to the strings $(0, 0^t)$ (i.e., sets r_i, τ_i to be the zero strings of appropriate length).
- **Phase II:** For each $i \in Q$, $\mathcal{S}$ plays out the execution of Γ honestly with the above sampled (r_i^R, τ_i^R). However, for each $i \notin Q$, it $\mathcal{S}$ runs the simulator $\mathcal{S}_\Gamma$ for Γ to simulate the view of $\mathcal{A}$ in this phase, and also extracts $\mathcal{A}$'s inputs in this phase, namely the strings $\{(s_i^0, s_i^1)\}_{i \notin Q}$.
- **Phase III:** $\mathcal{S}$ uses the weak parallel ε-simulator-extractor for ENMC to extract the value q_S from the sender-side non-malleable commitment. Note that such an extractor returns $\perp$ if even one of the component executions was declared invalid by the ENMC receiver. In such a scenario, the simulator calls for the ideal functionality $\mathcal{F}_{\mathrm{OT}}^n$ to abort the *entire* execution, and halts its own operation, outputting the adversary's state. Otherwise, it then sets $q_R = q \oplus q_S$ and sends this back to the sender. Next, when the corrupted sender opens its commitment to q_S, $\mathcal{S}$ opens its own commitments to $\{(r_i^R, \tau_i^R)\}_{i \in Q}$.
- **Phase IV:** $\mathcal{S}$ sends uniformly chosen bits $\{\alpha_j\}_{j \notin Q}$ to the corrupted sender, which sends strings (σ_0, σ_1) back in turn. $\mathcal{S}$ then computes $s_0 = \sigma_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and $s_1 = \sigma_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$ (recall that it extracted $\{(s_i^0, s_i^1)\}_{i \notin Q}$ earlier in **Phase II**). $\mathcal{S}$ then sends the functionality $\mathcal{F}_{\mathsf{OT}}^n$ for the relevant session with inputs (s_0, s_1). This completes its execution for this session.

Receiver corruption: $\mathcal{S}$ takes over the sender operation for this session. It proceeds as follows:

- **Phase I:** For $i \in [2\lambda]$, $\mathcal{S}$
  - uses the ε-simulator-extractor for ExtCom to extract the values (r_i^R, τ_i^R) committed by the corrupted receiver,
  - samples uniformly random strings (r_i, τ_i), and
  - sets $r_i^S = r_i \oplus r_i^R$, $\tau_i^S = \tau_i \oplus \tau_i^R$ and sends these values to the corrupted receiver.
- **Phase II:** $\mathcal{S}$ acts exactly as the honest sender does in this phase.
- **Phase III:** $\mathcal{S}$ acts exactly as the honest sender does in this phase.
- **Phase IV:** $\mathcal{S}$ computes an index $j^* \notin Q$ such that the values (r_{j^*}, τ_{j^*}) are consistent with the messages the corrupted receiver sent in the j^*th execution of Γ in **Phase II** (note that to perform this check, $\mathcal{S}$ crucially needs the values it extracted in **Phase I**).

If there exists no such session, $\mathcal{S}$ outputs a special symbol Fail and halts immediately. If the execution continues, it next receives the values $\{\alpha_j\}_{j\notin Q}$ from the corrupted receiver, and computes $r = \alpha_{j^*} \oplus r_{j^*}$ and queries $\mathcal{F}^n_{\mathsf{OT}}$ for the appropriate session with input r. Upon receiving a reply s_r from $\mathcal{F}^n_{\mathsf{OT}}$, $\mathcal{S}$ then computes the values (σ_0, σ_1) as follows:

- If $r = 0$, then it sets $\sigma_0 = s_0 \oplus (\bigoplus_{j\notin Q} s_j^{\alpha_j})$ and samples an uniform $\sigma_1 \xleftarrow{\$} \{0,1\}^\ell$.
- If $r = 1$, then it samples an uniform $\sigma_0 \xleftarrow{\$} \{0,1\}^\ell$ and sets $\sigma_1 = s_1 \oplus (\bigoplus_{j\notin Q} s_j^{1-\alpha_j})$.

Proof. Our proof relies on a simulator for the n-fold parallel execution of the scheme Π, which we can denote by Π^n. Our simulator $\mathcal{S}$ for this protocol is described in Algo. 11.1. It is easily seen that this simulator runs in (quantum) polynomial time.

Our proof, very broadly, rests primarily on two claims Lem. 27 and 28. The first is that the special abort condition Fail specified in the description of $\mathcal{S}$ is triggered with at most negligible probability. The second claim says that in the event that $\mathcal{S}$ does manage to not trigger Fail, it goes on to furnish a viable simulation of the execution Π^n. The logic of the proof is thus fairly straightforward, and we now turn to formalizing these claims and their respective justifications.

Lemma 27. *Denote the adversary for Π^n by $\mathcal{A}$. Recall that $\mathcal{S} = \mathcal{S}^{\mathcal{A}}$ is the simulator for Π^n (i.e., the procedure that produces $\mathsf{IDEAL}_{\mathcal{F}^n_{OT},\mathcal{S}^{\mathcal{A}}}(\lambda, \mathbf{x}, \rho_\lambda)$). Then we have*

$$\Pr[\mathsf{Fail} \leftarrow \mathcal{S}] \leq \mathsf{negl}(\lambda)$$

Proof. A similar proof already appears in [Wee10, Goy11]. The proof is in fact almost identical, with two notable differences: (1) we need to take care of the ε simulation error; (2) while [Wee10] defines non-malleability w.r.t. extractability and [Goy11] defines non-malleability w.r.t. replacement, we do not need such adjustments since we have fully many-many non-malleability.

Assume that there is in fact an adversary $\mathcal{A}$ for Π^n that runs in polynomial time, and also is such that $\Pr[\mathsf{Fail} \leftarrow \mathcal{S}] = \nu(\lambda)$ where $\nu(\cdot)$ is non-negligible. Assume that $\mathcal{S}$ outputs Fail in a specific session $k \in [n']$ (where $n' \coloneqq 2 \cdot \binom{n}{2}$).

We begin by examining exactly when $\mathcal{S}$ outputs Fail during simulation. From the description of $\mathcal{S}$, this happens only when the receiver is corrupted in session k, *and* there is *no* sub-index $j^* \in [2\lambda]$ which is (i) *not* opened in the cut and choose phase, and (ii) $\mathcal{A}$ behaves honestly in the session of Γ corresponding to j^*. In fact, with further consideration, we can infer that the following must also happen:

- For each pair in the 2λ executions, $\mathcal{A}$ behaves honestly in *exactly* one execution in the pair: if it did cheat in both, then it cannot succeed in opening either execution in the pair during the cut and choose execution; if indeed it did not cheat in either, we have nothing to worry about and can set j^* to be the index of the unopened execution. Consequently, there is an *unique* bit for every pair of (sub-)executions of Γ, and therefore a *unique* string $q^* \in \{0,1\}^\lambda$ across the 2λ executions of Γ, which when picked in the cut and choose phase will allow $\mathcal{A}$ to cheat.
- Further, $\mathcal{A}$ must ensure in Step 3 that q^* is indeed the result, i.e., it sends q_R such that $q_S \oplus q_R = q^*$.

We will use this observation to attack the non-malleability of ENMC and eventually derive a contradiction. To begin, we consider certain hybrids relating to Step 3 of Π^n. We define the view or output of a hybrid to be the adversary's view of the Π^n of the hybrid execution (therefore, corresponding to the output variable IDEAL for the simulator).

Hybrid H_0: This is simply the simulated execution of Π^n. For clearer contrast with subsequent hybrids, we make note of the index k identified as above. Further, we use $K' \subset [n']$ to denote the

indices of the sessions in which $\mathcal{A}$ corrupts the sender (note that k and K' are random variables). In particular, we emphasize that H_0 aborts the entire execution if any of the ENMC executions for Step 3a in the OT sessions in K' are not accepted by the corresponding receiver (see **Phase-III** for sender corruption in Algo. 11.1). Further, the hybrid also records the values $\{q_S^{k'}\}_{k' \in K'}$ that it extracts in Step 3a.

Remark 8. In H_0, we have that the following condition holds by the above analysis: with probability ν, the receiver picks q_R^k such that $q_S^k \oplus q_R^k = q^*$.

Hybrid H_1: This hybrid only differs from H_0 in the following operation: instead of using the simulator-extractor for ENMC to extract the values $\{q_S^{k'}\}_{k' \in K'}$ from Step 3a, it instead extracts them by running a *brute force attack* on the transcript of the Step 3a ENMC execution (recall that the hybrid then requires these values to finish its execution of Step 4). If however any of the ENMC sessions in K' corresponding to Step 3a are not completed successfully, the extracted values for *all* ENMC sessions in K' are set to be $\perp$.

Additionally, we make the following syntactic change for Step 3b in session k: instead of the hybrid sampling q_S^k directly, it instead samples an uniform value $q'^k_S \xleftarrow{\$} \{0,1\}^\lambda$, and then sets $q_S^k = q'^k_S$.

$\underline{\mathsf{Out}(H_0) \overset{\mathrm{s}}{\approx}_\varepsilon \mathsf{Out}(H_1)}$: A cursory examination reveals that the actual view of the adversary remains identical in both hybrids till Step 3. The extracted values $\{q_S^{k'}\}_{k' \in K'}$, and the resulting adversary state, are ε-indistinguishable in H_0 and H_1 by the (weak-parallel) ε-simulation-extraction guarantees of ENMC (as per Def. 4), and Step 4 is otherwise unaffected (crucially, note that both hybrids suspend the entire execution whenever any ENMC within OT sessions in K' is not accepted by the corresponding receiver, and the 'extracted' values are set to $\perp$ in such cases). Indistinguishability of the entire view of $\mathcal{A}$ follows.

Remark 9. We note from the above indistinguishability condition that in particular, the receiver in session k continues to pick q_R^k such that $q_S^k \oplus q_R^k = q^*$ with probability at least $\nu - \varepsilon$.

Hybrid H_2: This hybrid samples an uniform q'^k_S as in H_1, but sets $q_S^k = 0^\lambda$ instead of q'^k_S. All other steps remain the same as in H_1. We note here that the operation of H_2 is efficient *but* for the brute force extraction from Step 3a.

$\underline{\mathsf{Out}(H_1) \overset{\mathrm{c}}{\approx} \mathsf{Out}(H_2)}$: At first glance it may appear that this should be guaranteed directly by the hiding guarantee of the ENMC in Step 3a. This is however not the case: the reason is that while we have so far focused only on the session k, in truth the hybrid is also following the simulation strategy for corrupted *senders* in the sessions in K'. Note that in these sessions, the simulator itself is extracting the sender side ENMC values *using brute force* (starting from H_1)! This happens in parallel to the change we would make in H_2, and hence we cannot appeal to the computational hiding property.

What we can do however is appeal to the *weak 1-many non-malleability* property of ENMC. As pointed out, the executions of ENMC resemble that of a 1-many man-in-the-middle attack, where the one in session k is the 'left' session, and those in the sessions in K' form the 'right' sessions. Note that such a reduction, argued naively, will still inherit the brute-force extraction issue (whereas non-malleability is still a computational property). However, one can maneuver around this difficulty using the design of the non-malleability challenge: the extracted values $\{q_S^{k'}\}_{k' \in K'}$ (required by the

hybrid to complete the execution and manufacture its output) are not extracted by the adversary itself — instead, the challenger does so itself and then presents them to the distinguisher. Then, if all the right sessions are successfully completed, we allow the distinguisher (using a standard nonuniformity argument) to 'resurrect' the hybrid and complete the execution. Since the only brute force step by the hybrid is actually carried out by the challenger, this makes the adversary-distinguisher pair efficient, giving us a valid reduction to weak 1-many non-malleability.

Note that *weak* 1-many non-malleability indeed suffices in this setting, because of the deliberate construction of our hybrids. In more detail, observe that at the end of Step 3a, H_1 has the corresponding information to exactly reconstruct $\Gamma_{\{d_j\}_{j=1}^k}(\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(q'^k_S, \rho_\lambda))$, whereas H_2 has the information to exactly reconstruct $\Gamma_{\{d_j\}_{j=1}^k}(\mathsf{mim}[k]^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(0^\lambda, \rho_\lambda))$. In other words, any cases where the adversary chooses to selectively not complete some of the right ENMC sessions are discarded by the hybrids — and so only cases where all right sessions are successfully completed are considered. This ensures that we can then successfully rely on weak one-may non-malleability to argue similarity.

We proceed to formally reduce this claim to the weak 1-many nonmalleability of ENMC as defined in Def. 6. In more detail, assume there is a distinguisher D that can distinguish (with non-negligible advantage $\kappa(\lambda)$) between the outputs of H_2 and H_1. We describe a valid man-in-the-middle adversary and distinguisher pair $(\widetilde{A}, \widetilde{D})$ for the weak 1-many non-malleability game. We make use of the original adversary $\mathcal{A}$ and the assumed distinguisher D. The description of $\widetilde{A}$ and $\widetilde{D}$ are given below.

The adversary $\widetilde{\mathcal{A}}$ works as follows:

- Internally, it runs the hybrid H_2 with $\mathcal{A}$ embedded in the execution. It continues this up to the start of Step 3.
- Next, it externally begins participating in a 1-many challenge for ENMC, with challenge messages 0^λ and q'^k_S (uniformly sampling the latter).
- It forwards the challenger's sender (or 'left-side') messages as the Step 3a sender messages in session k of Π, and forwards out the receiver's session k replies out to the challenger as its own left-side receiver messages. Similarly, it cross-forwards the ENMC interactions for Step 3a in the sessions in K' to act as the 'right-side' ENMC interactions in the external challenge.
- At the end of Step 3a, it records its view and the adversary's state as its output, and halts. Note that the adversary' state may be quantum, but this is okay as the MIM adversary for post-quantum non-malleability is allowed to output quantum information.

The distinguisher $\widetilde{D}$ works as follows:

- $\widetilde{D}$ receives the output of $\widetilde{A}$ from the challenger, along with the left-side committed values q'^k_S. It then reconstructs the operation of the hybrid up to the start of Step 4 (along with the appropriate state of $\mathcal{A}$ at this stage).
- From the 1-many non-malleability challenger, it then receives the values committed by $\widetilde{\mathcal{A}}$ in the right side sessions, namely $\{q^{k'}_S\}_{k'\in K'}$.
- It then completes the execution of the hybrid and feeds the resulting hybrid output to D. It then outputs whatever D does.

It is easy to see that the operation of $\widetilde{A}$ and $\widetilde{D}$ follow exactly the execution of the hybrid H_1 if the ENMC challenger commits to q'^k_S in the left side commitment, and conversely these follow exactly the execution of H_2 when the ENMC challenger commits to 0^λ. Consequently, within $\widetilde{D}$, the view fed to the distinguisher D comes either from H_1 or H_2. Therefore $\widetilde{D}$ succeeds in winning the weak 1-many nonmalleability challenge whenever D distinguishes successfully within these hybrids.

Thus $(\widetilde{A}, \widetilde{D})$ win the weak 1-many non-malleability challenge with non-negligible probability κ, which contradicts the security of ENMC. We conclude that there is no such efficient distinguisher D that manages to distinguish between H_1 and H_2 with non-negligible probability.

Finally, in the hybrid H_2, we obtain a contradiction. Set $\varepsilon = \nu/2$. From the above, we can conclude that even in H_2, we have that the receiver in session k continues to pick q_R^k such that $q_S'^k \oplus q_R^k = q^*$ with non-negligible probability $\nu/2$. Recall that $q_S'^k$ is sampled uniformly by H_2 and not used in its internal execution of Π^n. Therefore, $q_S'^k$ is not seen by $\mathcal{A}$. This is a clear contradiction, since an uniformly sampled string of λ bits that is absent from the view of $\mathcal{A}$ has entropy λ from $\mathcal{A}$'s point of view; and so even a quantum (or even unbounded) machine can predict this string with probability $\frac{1}{2^\lambda}$, which is negligible.

This concludes the proof of Lem. 27.

□

We turn now to the second claim.

Lemma 28. *Conditioned on the event E where $\mathcal{S}$ does* not *output* Fail*, $\mathcal{S}$ produces a valid simulation of Π^n with respect to $\mathcal{A}$. Namely, given that E occurs, we have*

$$\mathsf{REAL}_{\Pi^n,\mathcal{A}}(\lambda, \mathbf{x}, \rho_\lambda) \overset{c}{\approx}_\varepsilon \mathsf{IDEAL}_{\mathcal{F}_{OT}^n,\mathcal{S}^{\mathcal{A}}}(\lambda, \mathbf{x}, \rho_\lambda)$$

Proof. This is essentially the same argument that is used in previous work [Wee10, IKLP06]. We refer the reader to these for a full proof, and include a sketch of the proof for the sake of completeness.

The case for simulating for a corrupted sender is straightforward, relying directly on the malicious sender security of Γ. The only thing of note here is that since the simulator relies on the simulator-extractor for ENMC, it inherits an ε simulation error for the portion of the view from Step 3a onwards. This is reflected in the final simulation guarantee stated in the lemma.

We focus instead on the corrupted receiver case. Here we observe that if $\mathcal{S}$ does not output Fail, then everything in the first three phases is essentially identical to how an honest sender acts, and it is only in the values (σ_0, σ_1) that the simulator's distribution changes. A distinguisher for the real and simulated view can then be used to attack the honest receiver security of Γ.

The attack is somewhat subtle, but it essentially uses the following observation: the 'hidden' value σ_{1-r} has the 'correct' distribution in the real execution, while in the simulated execution it is uniformly sampled. Thus a distinguisher for the real and simulated executions will be biased towards identifying an execution as 'real' when the hidden value has the correct distribution. The actual reduction makes an initial guess as to the hidden value and performs a sort of retroactive check, and uses the distinguisher's output cleverly to gain advantage in the honest receiver security game for Γ.

Of course, to make this reduction meaningful, care has to be taken to make sure that the internal Γ sub-execution within Π is indeed honest. This is essentially the function of the session j^* tracked by the simulator: for this session of Γ, we are guaranteed that the adversary uses honest inputs - as the simulator extracts all the receiver inputs initially and aborts if this is not the case.

Note that again we have to account for the ε-simulation guarantee from the extraction step in phase I, but this can be handled by a standard 'funneling' argument (i.e., start by assuming an distinguisher with a certain distinguishing advantage, and set ε to be significantly smaller to still derive a contradiction).

This concludes the proof of Lem. 28.

□

Finally, we tie together these claims to obtain the stated result. Since Fail is a simulator specific abort message, it is very easy to distinguish real and simulated executions whenever Fail is output. Now by Lem. 28, the maximum possible distinguishing advantage for a computationally bounded distinguisher (which we denoted by Δ_c) is ε whenever E occurs. Formally, we define $\Delta_c(X, Y) := \sup_{D \in \mathrm{PPT}} |\Pr[D(X) = 1] - \Pr[D(Y) = 1]|$. For brevity we will denote $\mathsf{REAL}_{\Pi^n, \mathcal{A}}(\lambda, \mathbf{x}, \rho_\lambda)$ by REAL and $\mathsf{IDEAL}_{\mathcal{F}_{\mathrm{OT}}^n, \mathcal{S}^{\mathcal{A}}}(\lambda, \mathbf{x}, \rho_\lambda)$ by IDEAL below.

We begin with the following decomposition

$$\begin{aligned}\Delta_c(\mathsf{REAL}, \mathsf{IDEAL}) =& [\Delta_c(\mathsf{REAL}, \mathsf{IDEAL})|E] \cdot \Pr[E] \\ &+ [\Delta_c(\mathsf{REAL}, \mathsf{IDEAL})|\overline{E}] \cdot \Pr\left[\overline{E}\right]\end{aligned}$$

By Lem. 27, we can write the RHS as

$$\begin{aligned}=& [\Delta_{D^*}(\mathsf{REAL}, \mathsf{IDEAL})|E] \cdot (1 - \mathsf{negl}(\lambda)) \\ &+ [\Delta_{D^*}(\mathsf{REAL}, \mathsf{IDEAL})|\overline{E}] \cdot \mathsf{negl}(\lambda),\end{aligned}$$

which is of course simply

$$\leq \varepsilon \cdot (1 - \mathsf{negl}(\lambda)) + 1 \cdot \mathsf{negl}(\lambda)$$

Which yields

$$\Delta_c(\mathsf{REAL}, \mathsf{IDEAL}) \leq \varepsilon' = \varepsilon + \mathsf{negl}(\lambda)$$

Readjusting $\varepsilon \to \varepsilon/2$ gives us the stated result.

This concludes the proof of Thm. 13.

□

12 Post-Quantum ε-Simulatable MPC

In this section we will describe and show security of a black-box, constant-round ε-MPC protocol. In fact we have gathered essentially all the ingredients needed for this task. The sole remaining component is the black-box compiler given in [IPS08]. Their protocol is a constant-round black-box MPC protocol, albeit with *UC security* and additionally assuming *ideal* OT channels. Here we will argue that this protocol when initialized with our malicious parallel OT protocol[37], will give us an MPC protocol with all the desired properties. We capture this in the following lemma.

Lemma 29. *The MPC protocol described in [IPS08], instantiated with the OT construction given in Prot. 4 (in lieu of an ideal OT functionality) is a* black-box, constant-round, post-quantum *ε-simulatable MPC protocol as defined in Def. 7.*

Proof. The construction simply involves instantiating the MPC protocol from [IPS08] using our OT scheme, as is already stated. We refer to their protocol as IPS for convenience.

We begin with a brief overview of the IPS MPC protocol. This involves composing *two* MPC protocols (titled the *inner* and *outer* protocols) in a specific fashion. The outer protocol uses the so-called *client-server* model, which involves parties called servers that have no input of their own but carry out the majority of the computation in the protocol. The key stratagem devised in IPS is to *emulate* the function of these servers distributedly using the inner MPC protocol. To ensure honesty, the protocol uses a mechanism introduced in IPS known as *watchlists*, which ensure that each party is able to monitor some emulated servers.

[37] Indeed, this observation has been employed in the classical setting [Wee10, Goy11].

Thus, in the running of the IPS protocol, there are various OT calls that are of two kinds. The first kind is used to initialize the watchlist mechanism, and this can be performed at the start of the protocol. The second kind is in the operation of the inner OT protocol that is used to emulate the servers (the inner protocol is in the OT-hybrid model, and needs to make calls to the ideal OT functionality).

We make the following two observations about the IPS protocol. These are easily verifiable from the descriptions present in [IPS08]. The first is that the watchlist setup can be initialized with an n-party OT functionality (this is observed in their work), i.e., $2 \cdot \binom{n}{2}$ OT calls in totally where each pair of parties run two OT calls with reversed role of sender and receiver. So we can use an n-fold parallel execution Prot. 4 in the beginning that suffices to setup the watchlists. The second is a *randomized OT trick* that can be used to 'prepone' the OT executions required by the inner protocol (this is also observed in their work). This modification is also needed for security. The idea is to basically initially perform n-fold OT executions with *random* values for the senders and receivers. Subsequently, the sender can send appropriately offset values (that encode OT inputs of its choice) to the receiver and the latter can recover its intended message from this.

This is to say that using this trick, the inner OT calls can also be pushed to the beginning of the protocol where we perform an n-fold parallel execution of Prot. 4 (for sufficiently long sender inputs). Therefore, we can complete an execution of the IPS protocol by beginning with two n-fold parallel (randomized) OT executions, for the watchlist and for the inner MPC protocol respectively. Subsequently, we proceed with the IPS protocol, setting up the watchlists and then executing the composed MPC protocol. Everytime the inner protocol would make an OT call, we use the random OT transformation and consume a predefined portion of the initial parallel randomized OT call to perform the actual OT interaction in the protocol.

Next we will argue why this achieves the desired security guarantee - security is already somewhat apparent and straightforward to establish, and we limit ourselves to addressing the more prominent concerns in this regard. We treat these in turn.

Constant-Round: The first thing to determine is simply whether the composed protocol is still constant round. While the total number of atomic OT calls made in the IPS protocol does depend on the number of parties (and hence grows polynomially with λ), these can be *batched* into a single parallel OT execution and shunted to the start of the protocol as described above. Now the IPS protocol itself is constant round (this includes the interactions made due to the randomized OT trick). In turn, our OT protocol from Prot. 4 also runs in constant rounds. The resulting protocol is therefore constant rounds. Indeed, this exact pipeline has been used in previous work on black-box MPC protocols to get protocols with *constant round overhead* (over the parallel OT part) in the classical standalone setting (see [Wee10, Goy11]).

Security: As pointed out, the security of the IPS protocol when initialized with a parallel OT protocol has been noted and employed in previous work ([Wee10, Goy11]). Our setting however presents two new challenges that are not present in the more standard setting, and we tackle them in turn.

Post-Quantum security: A simple examination of the IPS security proof and that of our OT protocol reveals that both of these are *black-box* and also *quantum compatible* - namely, they enjoy straightline simulation and are not reliant on classical rewinding. The combined security proof for the composed MPC protocol inherits these properties.

Sequential composition: A detail we have elided so far is that our OT is limited to ε-simulatability. This can affect the hybrids where we *sequentially* simulate various parallel OT executions in the

protocol. Fortunately, this exact kind of post-quantum sequential composition guarantee for ε-simulation has been shown in the work of [CCLY22a, Section 7.2].

□

13 Post-Quantum Equivocal Commitments

Here, we define equivocal commitments and construct an $\omega(1)$-round equivocal commitment scheme from OWFs. Looking ahead, this is used as a building block for constructing fully-simulatable (rather than ε-simulatable) extractable (batch) commitments in Sec. 14.

13.1 Definition

The definition of equivocality is given below. We stress that we require the simulation error to be negligible since this is used for achieving extractability with negligible simulation errors.

Definition 19 (PQ-EqCom). *A post-quantum commitment scheme $\langle C, R\rangle$ (as per Def. 1) is* equivocal *if there exists a QPT algorithm $\mathcal{SQ} = (\mathcal{SQ}_0, \mathcal{SQ}_1)$ (called the simulation equivocator) such that for any non-uniform QPT $R^*(\rho)$ and any polynomial $\ell(\cdot)$,*

$$\left\{(\mathsf{ST}_{R^*}, \mathsf{decom}) : \begin{array}{l}(\mathsf{ST}_{R^*}, \mathsf{ST}_{\mathcal{SQ}}) \leftarrow \mathcal{SQ}_0^{R^*(\rho)}(1^\lambda)\\ \mathsf{decom} \leftarrow \mathcal{SQ}_1(\mathsf{ST}_{\mathcal{SQ}}, m)\end{array}\right\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}}$$
$$\overset{c}{\approx}\left\{(\mathsf{ST}_{R^*}, \mathsf{decom}) : \begin{array}{l}(\tau, \mathsf{ST}_C, \mathsf{ST}_{R^*}) \leftarrow \langle C(m), R^*(\rho)\rangle(1^\lambda)\\ \mathsf{decom} \leftarrow C(\mathsf{ST}_C)\end{array}\right\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}}$$

The main theorem we prove in this section is the following:

Theorem 14. *Assuming the existence of post-quantum OWFs, there is a black-box construction of $\omega(1)$-round equivocal commitment schemes.*[38]

13.2 Construction with Noticeable Binding Error

For the ease of presentation, we start by constructing a *constant-round* equivocal commitment scheme wEqCom that has a noticeable binding error. Later, we argue that $\omega(1)$-times sequential repetition of wEqCom achieves negligible soundness error while preserving the equivocality. The construction is based on the ideas of [Kil88, Kil94], which have been used in many later works, e.g., [PW09, BCKM21]. The scheme wEqCom is described in Prot. 5 where it makes black-box use of a constant-round statistically-binding and computationally-hiding commitment scheme Com (e.g., Naor's commitment).

Protocol 5: Equivocal Commitments with Noticeable Soundness Error wEqCom

Parameters: Let $k = \Theta(\log\lambda)$ be a positive integer.

Inputs: Both parties receive λ as the common input. The committer in addition gets a string $m \in \{0,1\}^{\ell(\lambda)}$ as its private input where $\ell(\cdot)$ is a polynomial. For each $i \in [\ell]$, m_i denotes the i-th bit of m.

Commit Stage:

[38] Formally speaking, this means that for any time-constructible function $r(\cdot) = \omega(\lambda)$, there is a $r(\lambda)$-round equivocal commitment scheme. We use a similar convention throughout the paper.

1. C picks uniformly random bits $r_{i,j}$ and defines

$$\begin{pmatrix} s_{i,j}^{00} & s_{i,j}^{01} \\ s_{i,j}^{10} & s_{i,j}^{11} \end{pmatrix} = \begin{pmatrix} r_{i,j} & m_i \oplus r_{i,j} \\ r_{i,j} & m_i \oplus r_{i,j} \end{pmatrix}$$

for $i \in [\ell]$ and $j \in [k]$.

2. C commits to $\{s_{i,j}^{ab}\}_{i\in[\ell],j\in[k],(a,b)\in\{0,1\}^2}$ using Com in a bit-by-bit manner in parallel. Let $\{\tau_{i,j}^{ab}\}_{i\in[\ell],j\in[k],(a,b)\in\{0,1\}^2}$ be the corresponding transcripts.
3. R randomly picks uniformly random bits c_j for $j \in [k]$ and sends them to C.
4. C reveals $s_{i,j}^{0c_j}$ and $s_{i,j}^{1c_j}$ along with the corresponding decommit information w.r.t. Com in Step 2 to R for $i \in [\ell]$ and $j \in [k]$.
5. R accepts if all the decommit information are valid and $s_{i,j}^{0c_j} = s_{i,j}^{1c_j}$ for all $i \in [\ell]$ and $j \in [k]$ and otherwise rejects.

Decommit Stage:

1. C picks uniformly random bits $d_{i,j}$ and reveals m and $s_{i,j}^{d_{i,j}(1-c_j)}$ along with the corresponding decommit information w.r.t. Com in Step 2 of Commit Stage to R for $i \in [\ell]$ and $j \in [k]$.
2. R accepts if all the decommit information are valid and $s_{i,j}^{d_{i,j}(1-c_j)} = m_i \oplus s_{i,j}^{d_{i,j}c_j}$ for all $i \in [\ell]$ and $j \in [k]$ and otherwise rejects. (Note that $s_{i,j}^{d_{i,j}c_j}$ is already revealed in the commit stage.)

Theorem 15. *The scheme* wEqCom *(Prot. 5) is constant-round and satisfies computational hiding, equivocality (as per Def. 19), and* $(2^{-k} + \mathsf{negl}(\lambda))$*-statistical binding, which is defined similarly to statistical binding (as per Def. 1) except that we only require the malicious committer's winning probability to be at most* $2^{-k} + \mathsf{negl}(\lambda)$ *instead of* $\mathsf{negl}(\lambda)$*.*

Proof. It is clear from the description that it is constant-round. Since computatioanl hiding immediately follows from equivocality, we prove $(2^{-k} + \mathsf{negl}(\lambda))$-statistical binding and equivocality below.

$(2^{-k} + \mathsf{negl}(\lambda))$-Statistical Binding. In an execution of the protocol between unbounded-time malicious committer C^* and honest receiver R, let Bad be the event that any of $\tau_{i,j}^{ab}$ can be decommitted to more than one messages. By statistical binding of Com, Bad occurs with a negligible probability. Below, we assume that Bad does not occur. For $i \in [\ell]$, let $\mathsf{Inconsistent}_i$ be the event that for all $j \in [k]$, there exists $b_{i,j}$ such that $\mathsf{val}(\tau_{i,j}^{0b_{i,j}}) \neq \mathsf{val}(\tau_{i,j}^{1b_{i,j}})$. When $\mathsf{Inconsistent}_i$ occurs, then C^* can pass the verification by R in the commit stage only if $c_j = 1 - b_{i,j}$ for all $j \in [k]$. Since c_j is uniformly random, this occurs with probability 2^{-k}. Thus, whenever C^* passes the verification in the commit stage, then neither of Bad or $\mathsf{Inconsistent}_i$ for any $i \in [k]$ occurs except for probability $2^{-k} + \mathsf{negl}(\lambda)$. When neither of Bad or $\mathsf{Inconsistent}_i$ occurs, there is j_i^* such that $\mathsf{val}(\tau_{i,j_i^*}^{00}) = \mathsf{val}(\tau_{i,j_i^*}^{10})$ and $\mathsf{val}(\tau_{i,j_i^*}^{01}) = \mathsf{val}(\tau_{i,j_i^*}^{11})$, in which case the i-th bit can be only decommitted to $\mathsf{val}(\tau_{i,j_i^*}^{00}) \oplus \mathsf{val}(\tau_{i,j_i^*}^{01}) = \mathsf{val}(\tau_{i,j_i^*}^{10}) \oplus \mathsf{val}(\tau_{i,j_i^*}^{11})$.[39] Thus, except for probability $2^{-k} + \mathsf{negl}(\lambda)$, all the bits can be decommitted to either of 0 or 1. This means that it satisfies $(2^{-k} + \mathsf{negl}(\lambda))$-statistical binding.

Equivocality. The proof strategy is similar to that in the security proof of the equivocality compiler of [BCKM21], which in turn is based on quantum zero-knowledge proofs by Watrous [Wat09]. We

[39] We define $\bot \oplus \beta = \bot$ for $\beta \in \{0, 1, \bot\}$.

first construct a weaker simulation equivocator that guesses the challenges and works only when the guess is correct. Then we compile it to the full-fledged simulation equivocator (as per Def. 19) by using Watrous' rewinding lemma (Lem. 6).

First, we consider the following algorithm Q:

$\mathsf{Q}^{R^*(\rho)}(1^\lambda)$:

1. Randomly pick bits c'_j for $j \in [k]$ and $e_{i,j}$ and $r_{i,j}$ for $i \in [\ell]$ and $j \in [k]$.
2. Define
$$\begin{pmatrix} s_{i,j}^{0c'_j} & s_{i,j}^{0(1-c'_j)} \\ s_{i,j}^{1c'_j} & s_{i,j}^{1(1-c'_j)} \end{pmatrix} = \begin{pmatrix} r_{i,j} & e_{i,j} \\ r_{i,j} & 1-e_{i,j} \end{pmatrix}$$
for $i \in [\ell]$ and $j \in [k]$.
3. Commit to $\{s_{i,j}^{ab}\}_{i\in[\ell],j\in[k],(a,b)\in\{0,1\}^2}$ using Com in a bit-by-bit manner in parallel. Let $\{\tau_{i,j}^{ab}\}_{i\in[\ell],j\in[k],(a,b)\in\{0,1\}^2}$ be the corresponding transcripts. Send them to the malicious receiver R^*.
4. Receive bits $\{c_j\}_{j\in[k]}$ from R^*.
5. If $c_j \neq c'_j$ for some $j \in [k]$, output $\beta = 1$ and immediately halt. Otherwise, proceed to the next step.
6. Reveal $s_{i,j}^{0c_j} = s_{i,j}^{1c_j} = r_{i,j}$ along with the corresponding decommit information w.r.t. Com to R for $i \in [\ell]$ and $j \in [k]$.
7. Define $\mathsf{decom}_{i,j}^\eta$ to be the decommit information for $\tau_{i,j}^{(e_{i,j}\oplus\eta)(1-c_j)}$ w.r.t. Com for $i \in [\ell]$, $j \in [k]$, and $\eta \in \{0,1\}$. Note that the committed message in $\tau_{i,j}^{(e_{i,j}\oplus\eta)(1-c_j)}$ is η.
8. Output a quantum state σ that consists of the final state ST_{R^*} of R^* and $\{\mathsf{decom}_{i,j}^\eta\}_{i\in[\ell],j\in[k],\eta\in\{0,1\}}$ along with a bit $\beta = 0$.

Let Q_ρ^0 be the distribution of σ output by $\mathsf{Q}^{R^*(\rho)}(1^\lambda)$ conditioned on that $\beta = 0$. Let $p(\rho)$ be the probability that $\mathsf{Q}^{R^*(\rho)}(1^\lambda)$ returns $\beta = 0$. By computational hiding of Com, it is easy to see that we have
$$|p(\rho) - 2^{-k}| \leq \mathsf{negl}(\lambda)$$
for any quantum advice ρ. Thus, by applying Watrous' rewinding lemma (Lem. 6) with $p_0 = 2^{-k} - \mathsf{negl}(\lambda)$, $q = 2^{-k}$, $\gamma = \mathsf{negl}(\lambda)$, and $T = \lfloor \frac{\log(1/\gamma)}{4p_0(1-p_0)} \rfloor$, we obtain a QPT algorithm $\widetilde{\mathsf{Q}}$ that makes black-box use of $R^*(\rho)$ such that
$$\mathsf{TD}(\mathsf{Q}_\rho^0, \widetilde{\mathsf{Q}}^{R^*(\rho)}(1^\lambda)) \leq 4\sqrt{\gamma}\frac{\log(1/\gamma)}{p_0(1-p_0)} = \mathsf{negl}(\lambda) \tag{59}$$
where we used $k = \Theta(\log\lambda)$ and thus $p_0 = 2^{-k} - \mathsf{negl}(\lambda) = 1/\mathsf{poly}(\lambda)$. We remark that $\widetilde{\mathsf{Q}}$ plays the role of R in Lem. 6. We changed the notation to avoid confusion with the malicious receiver R^*. We also remark that $\widetilde{\mathsf{Q}}$ makes black-box use of $R^*(\rho)$ since it makes black-box use of Q, which in turn makes black-box use of $R^*(\rho)$.

We are now ready to describe the simulation equivocator $\mathcal{SQ} = (\mathcal{SQ}_0, \mathcal{SQ}_1)$:

$\mathcal{SQ}_0^{R^*(\rho)}(1^\lambda)$:

1. Run $\widetilde{\mathsf{Q}}^{R^*(\rho)}(1^\lambda)$ to obtain ST_{R^*} and $\{\mathsf{decom}_{i,j}^\eta\}_{i\in[\ell],j\in[k],\eta\in\{0,1\}}$.
2. Output ST_{R^*} and $\mathsf{ST}_{\mathcal{SQ}} := \{\mathsf{decom}_{i,j}^\eta\}_{i\in[\ell],j\in[k],\eta\in\{0,1\}}$.

$\mathcal{SQ}_1(\mathsf{ST}_{\mathcal{SQ}}, m)$:

1. Parse $\mathsf{ST}_{\mathcal{SQ}} = \{\mathsf{decom}^{\eta}_{i,j}\}_{i\in[\ell],j\in[k],\eta\in\{0,1\}}$.
2. Let m_i be the i-th bit of m for $i \in [\ell]$.
3. Output $\mathsf{decom} := \{\mathsf{decom}^{m_i}_{i,j}\}_{i\in[\ell],j\in[k]}$.

Let $\overline{\mathcal{SQ}}_0^{R^*(\rho)}(1^\lambda)$ be a not necessarily QPT algorithm that works similarly to $\mathcal{SQ}_0^{R^*(\rho)}(1^\lambda)$ except that it samples ST_{R^*} and $\{\mathsf{decom}^{\eta}_{i,j}\}_{i\in[\ell],j\in[k],\eta\in\{0,1\}}$ from Q^0_ρ. By Eq. (59), we have

$$\left\{(\mathsf{ST}_{R^*},\mathsf{decom}) : \begin{array}{l}(\mathsf{ST}_{R^*},\mathsf{ST}_{\mathcal{SQ}}) \leftarrow \mathcal{SQ}_0^{R^*(\rho)}(1^\lambda)\\ \mathsf{decom} \leftarrow \mathcal{SQ}_1(\mathsf{ST}_{\mathcal{SQ}},m)\end{array}\right\}_{\lambda\in\mathbb{N},m\in\{0,1\}^{\ell(\lambda)}}$$
$$\stackrel{\mathrm{s}}{\approx}\left\{(\mathsf{ST}_{R^*},\mathsf{decom}) : \begin{array}{l}(\mathsf{ST}_{R^*},\mathsf{ST}_{\mathcal{SQ}}) \leftarrow \overline{\mathcal{SQ}}_0^{R^*(\rho)}(1^\lambda)\\ \mathsf{decom} \leftarrow \mathcal{SQ}_1(\mathsf{ST}_{\mathcal{SQ}},m)\end{array}\right\}_{\lambda\in\mathbb{N},m\in\{0,1\}^{\ell(\lambda)}}.$$

Moreover, by computational hiding of Com, it is easy to show that

$$\left\{(\mathsf{ST}_{R^*},\mathsf{decom}) : \begin{array}{l}(\mathsf{ST}_{R^*},\mathsf{ST}_{\mathcal{SQ}}) \leftarrow \overline{\mathcal{SQ}}_0^{R^*(\rho)}(1^\lambda)\\ \mathsf{decom} \leftarrow \mathcal{SQ}_1(\mathsf{ST}_{\mathcal{SQ}},m)\end{array}\right\}_{\lambda\in\mathbb{N},m\in\{0,1\}^{\ell(\lambda)}}$$
$$\stackrel{\mathrm{c}}{\approx}\left\{(\mathsf{ST}_{R^*},\mathsf{decom}) : \begin{array}{l}(\tau,\mathsf{ST}_C,\mathsf{ST}_{R^*}) \leftarrow \langle C(m), R^*(\rho)\rangle(1^\lambda)\\ \mathsf{decom} \leftarrow C(\mathsf{ST}_C)\end{array}\right\}_{\lambda\in\mathbb{N},m\in\{0,1\}^{\ell(\lambda)}}.$$

Combining the above, the proof of equivocality is completed. □

13.3 Reducing Binding Error

We show that sequential repetition of wEqCom (Prot. 5) reduces the binding error to be negligible while preserving equivocality.

Protocol 6: Equivocal Commitments EqCom

Parameters: Let $n = \omega(1)$ be a positive integer.
Inputs: Both parties receive λ as the common input. The committer in addition gets a string $m \in \{0,1\}^{\ell(\lambda)}$ as its private input where $\ell(\cdot)$ is a polynomial.

Commit Stage:

1. C commits to m using wEqCom n times in a sequential manner.

Decommit Stage:

1. C reveals m along with the corresponding decommit information w.r.t. all the n executions of wEqCom.
2. R accepts if all the decommit information are valid and otherwise rejects.

Theorem 16. *The scheme* EqCom *(Prot. 6) satisfies computational hiding, equivocality (as per Def. 19), and statistical binding.*

Proof. Statistical binding follows from $(2^{-k}+\mathsf{negl}(\lambda))$-statistical binding of wEqCom noting that the binding error is exponentially reduced by sequential repetition and $(2^{-k}+\mathsf{negl}(\lambda))^n = \mathsf{negl}(\lambda)$ when $k = \Theta(\log\lambda)$ and $n = \omega(1)$. Equivocality immediately follows from that of wEqCom noting that

equivocality is preserved under sequential composition. Indeed, this can be shown by a straightforward hybrid argument (see e.g., [BCKM21]). Computational hiding immediately follows from equivocality. □

Since EqCom runs in $\omega(1)$ rounds makes black-box use of OWFs, Thm. 16 implies Thm. 14.

14 Post-Quantum Extractable Batch Commitments

14.1 Definitions

Definition 20 (Post-Quantum Batch Commitments). *A* post-quantum batch commitment scheme $\langle C, R\rangle$ *is a classical interactive protocol between interactive* PPT *machines* C *and* R*. Let* $\mathbf{m} = (m_1, ..., m_n) \in \{0,1\}^{\ell(\lambda)\times n(\lambda)}$ *(where* $\ell(\cdot)$ *and* $n(\lambda)$ *are some polynomials) be a sequence of messages that* C *wants to commit to. The protocol consists of the following stages:*

- **Commit Stage:** $C(\mathbf{m})$ *and* R *interact with each other to generate a transcript (which is also called a commitment) denoted by* τ,[40] C*'s state* ST_C*, and* R*'s output* $b_{\mathrm{com}} \in \{\bot, \top\}$ *indicating acceptance (i.e.,* $b_{\mathrm{com}} = \top$*) or rejection (i.e.,* $b_{\mathrm{com}} = \bot$*). We denote this execution by* $(\tau, \mathsf{ST}_C, b_{\mathrm{com}}) \leftarrow \langle C(m), R\rangle(1^\lambda)$*. 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 sequence of decommitments* $\mathsf{decom} = (\mathsf{decom}_1, ..., \mathsf{decom}_n)$ *from* ST_C*. We denote this procedure by* $\mathsf{decom} \leftarrow C(\mathsf{ST}_C)$*. Then it sends a sequence of messages* $\mathbf{m} = (m_1, ..., m_n)$ *and the sequence of decommitments* $\mathsf{decom} = (\mathsf{decom}_1, ..., \mathsf{decom}_n)$ *to* R*. For each* $i \in [n]$*,* R *runs a deterministic verification procedure* $b_{\mathrm{dec},i} \leftarrow \mathsf{Verify}_i(\tau, m_i, \mathsf{decom}_i)$ *where* $b_{\mathrm{dec},i} = \top$ *and* $b_{\mathrm{dec},i} = \bot$ *indicate acceptance and rejection on the* i*-th bit, respectively. W.l.o.g., we assume that* R *always rejects (i.e.,* $\mathsf{Verify}_i(\tau, \cdot, \cdot) = \bot$ *for all* $i \in [n]$*) whenever* $b_{\mathrm{com}} = \bot$*. (Note that w.l.o.g.,* τ *can include* b_{com} *because we can always modify the protocol to ask* R *to send* b_{com} *as the last round message.)*

The scheme satisfies the following requirements:

1. **(Completeness.)** *For any polynomials* $\ell : \mathbb{N} \to \mathbb{N}$ *and* $n : \mathbb{N} \to \mathbb{N}$*, any* $\mathbf{m} \in \{0,1\}^{\ell(\lambda)\times n(\lambda)}$*, and any* $i \in [n]$*, it holds that*

$$\Pr\left[b_{\mathrm{com}} = b_{\mathrm{dec},i} = \top : \begin{array}{c}(\tau, \mathsf{ST}_C, b_{\mathrm{com}}) \leftarrow \langle C(\mathbf{m}), R\rangle(1^\lambda)\\ (\mathsf{decom}_1, ..., \mathsf{decom}_n) \leftarrow C(\mathsf{ST}_C)\\ b_{\mathrm{dec},i} \leftarrow \mathsf{Verify}_i(\tau, m_i, \mathsf{decom}_i)\end{array}\right] = 1.$$

2. **(Statistically binding.)** *For any unbounded-time committer* C^**, the following holds:*

$$\Pr\left[\begin{array}{c}\exists\ i \in [n], m_0, m_1, \mathsf{decom}_0, \mathsf{decom}_1,\ s.t.\ m_0 \neq m_1 \wedge\\ \mathsf{Verify}_i(\tau, m_0, \mathsf{decom}_0) = \mathsf{Verify}_i(\tau, m_1, \mathsf{decom}_1) = \top\end{array} : (\tau, \mathsf{ST}_{C^*}, b_{\mathrm{com}}) \leftarrow \langle C^*, R\rangle(1^\lambda)\right] = \mathsf{negl}(\lambda).$$

3. **(Computationally Hiding.)** *For any non-uniform QPT receiver* R^* *and any polynomials* $\ell : \mathbb{N} \to \mathbb{N}$ *and* $n : \mathbb{N} \to \mathbb{N}$ *, the following holds:*

$$\left\{\mathsf{OUT}_{R^*}\langle C(\mathbf{m}_0), R^*\rangle(1^\lambda), \{\mathsf{decom}_{i'}\}_{i'\neq i}\right\}_{\lambda\in\mathbb{N},\ i\in[n(\lambda)],\ \mathbf{m}_0,\mathbf{m}_1\in\Delta_i}$$

[40] That is, we regard the whole transcript as a commitment.

$$\overset{c}{\approx} \left\{\mathsf{OUT}_{R^*}\langle C(\mathbf{m}_1), R^*\rangle(1^\lambda), \{\mathsf{decom}_{i'}\}_{i'\neq i}\right\}_{\lambda\in\mathbb{N},\ i\in[n(\lambda)],\ \mathbf{m}_0,\mathbf{m}_1\in\Delta_i},$$

where

$$\Delta_i := \{(\mathbf{m}_0 = (m_{0,1},...,m_{0,n}), \mathbf{m}_1 = (m_{1,1},...,m_{1,n})) \in (\{0,1\}^{\ell(\lambda)})^2 : \forall i' \in [n]\setminus\{i\}\ m_{0,i'} = m_{1,i'}\},$$

$\mathsf{OUT}_{R^*}\langle C(\mathbf{m}_b), R^*\rangle(1^\lambda)$ $(b\in\{0,1\})$ *denotes the output of* R^* *at the end of the commit stage, and* $(\mathsf{decom}_1,...,\mathsf{decom}_n)$ *denotes the sequence of decommitments generated by* C *in the decommit stage.*

Remark 10. By a straightforward hybrid argument, the above computational hiding property implies the following:

– **(Computationally Hiding w.r.t. Subsets.)** For any non-uniform QPT receiver R^* and any polynomials $\ell:\mathbb{N}\to\mathbb{N}$ and $n:\mathbb{N}\to\mathbb{N}$, the following holds:

$$\begin{aligned}&\left\{\mathsf{OUT}_{R^*}\langle C(\mathbf{m}_0), R^*\rangle(1^\lambda), \{\mathsf{decom}_{i'}\}_{i'\notin S}\right\}_{\lambda\in\mathbb{N},\ S\subseteq[n(\lambda)],\ \mathbf{m}_0,\mathbf{m}_1\in\Delta_S}\\ \overset{c}{\approx}\ &\left\{\mathsf{OUT}_{R^*}\langle C(\mathbf{m}_1), R^*\rangle(1^\lambda), \{\mathsf{decom}_{i'}\}_{i'\notin S}\right\}_{\lambda\in\mathbb{N},\ S\subseteq[n(\lambda)],\ \mathbf{m}_0,\mathbf{m}_1\in\Delta_S},\end{aligned}$$

where

$$\Delta_S := \{(\mathbf{m}_0 = (m_{0,1},...,m_{0,n}), \mathbf{m}_1 = (m_{1,1},...,m_{1,n})) \in (\{0,1\}^{\ell(\lambda)})^2 : \forall i' \in [n]\setminus S\ m_{0,i'} = m_{1,i'}\},$$

$\mathsf{OUT}_{R^*}\langle C(\mathbf{m}_b), R^*\rangle(1^\lambda)$ $(b\in\{0,1\})$ denotes the output of R^* at the end of the commit stage, and $(\mathsf{decom}_1,...,\mathsf{decom}_n)$ denotes the sequence of decommitments generated by C in the decommit stage.

Similarly to the stand-alone setting (Def. 2), we define committed values for batch commitments as follows.

Definition 21 (Committed Values for Batch Commitments). *For a statistically binding batch commitment scheme* $\langle C, R\rangle$ *(as per Def. 20), we define the value function as follows:*

$$\mathsf{val}_i(\tau) := \begin{cases} m_i & \text{if } \exists \text{ unique } m_i \text{ s.t. } \exists\ \mathsf{decom}_i, \mathsf{Verify}_i(\tau, m_i, \mathsf{decom}_i) = 1 \\ \bot & \text{otherwise}\end{cases},$$

where Verify_i *is as defined in Def. 20.*

Then we define extractability for batch commitments. This is a natural extension of that in the stand-alone setting (Def. 3) but there is a crucial difference that we require simulation with negligible errors instead of ε-simulation because the purpose of this section is to achieve negligible simulation errors at the cost of sacrificing round complexity.

Definition 22 (PQ-ExtBCom). *A post-quantum batch commitment scheme* $\langle C, R\rangle$ *(as per Def. 20) is* extractable *if there exists a QPT algorithm* $\mathcal{SE}$ *(called the simulation extractor) such that for any non-uniform QPT* $C^*(\rho)$*,*

$$\left\{\mathcal{SE}^{C^*(\rho)}(1^\lambda)\right\}_\lambda \overset{c}{\approx} \left\{(\{\mathsf{val}_i(\tau)\}_{i\in[n]}, \mathsf{ST}_{C^*}) : (\tau, \mathsf{ST}_{C^*}, b_{\mathrm{com}}) \leftarrow \langle C^*(\rho), R\rangle(1^\lambda)\right\}_\lambda,$$

where $\mathsf{val}_i(\tau)$ *is the value committed by* C^* *as defined in Def. 21.*

Remark 11. We remark that we only require computational indistinguishability between the simulated and real execution while Def. 3 requires statistical indistinguishability (with a noticeable simulation error). This is because computational indistinguishability is sufficient for our purpose.

Remark 12. One may find a conceptual similarity between extractable batch commitments (Def. 22) and parallelly extractable commitments (Def. 4). Indeed, if a (stand-alone) commitment scheme (as per Def. 1) satisfies the negligible simulation error version of parallel extractability (as per Def. 4), then its parallel composition is an extractable batch commitment scheme (as per Def. 22). However, we do not know how to construct a commitment scheme that satisfies the negligible simulation error version of parallel extractability. This is why we introduced extractable batch commitments.

As an intermediate tool towards constructing extractable batch commitments (as per Def. 22), we introduce a weaker security notion which we call *extractability with over-extraction*. Intuitively, it is similar to the full-fledged extractability (Def. 22) except that we allow the simulation extractor to extract non-$\bot$ messages even if the transcript is invalid (i.e., there is no accepting decommitment).

Definition 23 (PQ-ExtBCom with Over-extraction). *A post-quantum batch commitment scheme $\langle C, R\rangle$ (as per Def. 20) is* extractable with over-extraction *if there exists a QPT algorithm $\mathcal{SE}_{\text{over}}$ (called the simulation extractor with over-extraction) such that for any non-uniform QPT $C^*(\rho)$,*

$$\left\{(\tau, \mathsf{ST}_{C^*}) : (\tau, \mathsf{ST}_{C^*}, \{m_{\mathsf{Ext},i}\}_{i\in[n]}) \leftarrow \mathcal{SE}_{\text{over}}^{C^*(\rho)}(1^\lambda)\right\}_\lambda \overset{c}{\approx} \left\{(\tau, \mathsf{ST}_{C^*}) : (\tau, \mathsf{ST}_{C^*}, b_{\text{com}}) \leftarrow \langle C^*(\rho), R\rangle(1^\lambda)\right\}_\lambda,$$

and

$$\Pr\left[\exists i \in [n] \ s.t. \ \mathsf{val}_i(\tau) \notin \{m_{\mathsf{Ext},i}, \bot\}\right] \leq \mathsf{negl}(\lambda)$$

where $(\tau, \mathsf{ST}_{C^*}, \{m_{\mathsf{Ext},i}\}_{i\in[n]}) \leftarrow \mathcal{SE}_{\text{over}}^{C^*(\rho)}(1^\lambda)$ and $\mathsf{val}_i(\mathsf{com})$ is the value committed by C^* as defined in Def. 21.

Remark 13. Extractability with over-extraction is conceptually similar to weak extractability defined in [CCLY22a] in the sense that both only require the extracted message be correct only when the transcript is valid. However, the crucial difference is that extractability with over-extraction requires that simulation of the committer's state be indistinguishable from the real one even when the transcript is invalid whereas weak extractability only requires it when the transcript is valid.

14.2 Extractable Batch Commitments with Over-extraction

Our construction of an extractable batch commitment scheme with over-extraction is described in Prot. 7. It makes black-box use of the following building blocks:

1. A constant-round statistically-binding, computationally-hiding commitment Com, (e.g., Naor's commitment).
2. A parallel oblivious transfer protocol OT that satisfies ε-simulation security against malicious receivers and indistinguishability-based security against malicious senders. Assuming $t(\lambda)$-round semi-honest OT, [CCLY22a] gives a black-box and $O(t(\lambda))$-round construction of such a protocol.[41]
3. An equivocal commitment scheme EqCom (as per Def. 19). A $\omega(1)$-round construction of it is known assuming only black-box access to post-quantum secure OWFs (Thm. 14).

[41] Their construction is shown to satisfy ε-simulation security for both malicious senders and malicious receivers, and ε-simulation security immediately implies indistinguishability-based security.

Protocol 7: Extractable Batch Commitments with Over-Extraction $\mathsf{OverExtBCom}$

Parameters: Let $k = \omega(\log \lambda)$ be a positive integer. We use k to mean the number of parallel sessions in OT.

Inputs: Both parties receive λ as the common input. The committer in addition gets a sequence of strings $\mathbf{m} = (m_1, ..., m_n) \in \{0,1\}^{\ell(\lambda) \times n(\lambda)}$ as its private input where $\ell(\cdot)$ and $n(\cdot)$ are polynomials.

Commit Stage:

1. C commits to m_i using Com for all $i \in [n]$ in parallel. Let com_i be the transcript of the i-th execution.
2. For $i \in [n]$, C generates $2k$-out-of-$2k$ XOR secret sharing $\{s^b_{i,j}\}_{j\in[k],b\in\{0,1\}}$ of m_i. That is, they are uniformly random under the constraint that $\bigoplus_{j\in[k],b\in\{0,1\}} s^b_{i,j} = m_i$.
3. C and R execute n-parallel executions of OT.[a] We refer to the i-th execution by OT_i where C uses $\{s^0_{i,j}, s^1_{i,j}\}_{j\in[k],b\in\{0,1\}}$ as input and R uses independently and uniformly random bits $\{r_{i,j}\}_{j\in[k]}$ as input.
4. C and R now engage in the following coin-flipping subprotocol as detailed below.
   (a) R samples a random string $\theta_R \leftarrow \{0,1\}^{nk}$ and commits to it using EqCom.
   (b) C samples a random string $\theta_C \leftarrow \{0,1\}^{nk}$ and sends it to R.
   (c) R sends to C the value θ_R together with the corresponding decommitment information w.r.t. the EqCom in Step 4a. Now, C and R agree on a random string $\theta := \theta_R \oplus \theta_C \in \{0,1\}^{nk}$. Interpret θ as a family $\{t_{i,j}\}_{i\in[n],j\in[k]}$ of bits. That is, let $t_{i,j}$ be the $(i-1)k+j$-th bit of θ for $i \in [n]$ and $j \in [k]$.
5. C sends $s^{t_{i,j}}_{i,j}$ to R for $i \in [n]$ and $j \in [k]$.
6. R never rejects in the commit stage, i.e., it always outputs $b_{\mathrm{com}} = \top$. C sets the randomness used in the commit stage as ST_C and keep it for the decommit stage.

Decommit Stage:

1. For $i \in [n]$, C defines decom_i to be a string consisting of the decommit information of com_i w.r.t. Com in Step 1 of the Commit Stage, $\{s^0_{i,j}, s^1_{i,j}\}_{j\in[k],b\in\{0,1\}}$, and the sender's randomness used in OT_i in Step 3 of the Commit Stage.
2. C sends $\mathbf{m} = (m_1, ..., m_n)$ along with $\mathsf{decom} = (\mathsf{decom}_1, ..., \mathsf{decom}_n)$ to R.
3. For each $i \in [n]$, R runs the verification procedure Verify_i that accepts if
   (a) the decommitment of com_i is valid w.r.t. the committed message m_i,
   (b) $\bigoplus_{j\in[k],b\in\{0,1\}} s^b_{i,j} = m_i$,
   (c) the revealed randomness for OT_i is consistent to the transcript of OT_i with input $\{s^0_{i,j}, s^1_{i,j}\}_{j\in[k],b\in\{0,1\}}$, and
   (d) the string sent in Step 5 of the Commit Stage is consistent to $\{s^0_{i,j}, s^1_{i,j}\}_{j\in[k],b\in\{0,1\}}$, i.e., it is equal to $s^{t_{i,j}}_{i,j}$ where $t_{i,j}$ is the bit generated in Step 4 of the Commit Stage.

   R accepts if Verify_i accepts for all $i \in [n]$ and otherwise rejects.

[a] Note that OT itself is a k-parallel OT, and thus nk-parallel executions are happening in total.

Security. The security of Prot. 7 is stated as the following theorem.

Theorem 17. *Assuming the existence of $t(\lambda)$-round semi-honest OT protocols, there exists (i.e., Prot. 7) a black-box, $O(t(\lambda))+\omega(1)$-round construction of batch commitments that satisfies statistical binding, computational hiding (as per Def. 20), and extractability with over-extraction (as per Def. 23).*

Proof. Statistical binding property immediately follows from that of Com. Below, we show computational hiding and extractability with over-extraction.

Computational Hiding. Let $R^*(\rho)$ be any malicious QPT receiver. Fix $i^* \in [\ell]$ and pair of sequence of messages $\mathbf{m}_0$ and $\mathbf{m}_1$ that differ only on the i^*-th component. We consider the following hybrids for $b \in \{0,1\}$ and noticeable function ε.

Hybrid H_b: This hybrid simulates execution between C with input $\mathbf{m}_b$ and $R^*(\rho)$ and outputs $(\mathsf{OUT}_{R^*}\langle C(\mathbf{m}_b), R^*\rangle(1^\lambda), \{\mathsf{decom}_i\}_{i\neq i^*})$ where $(\mathsf{decom}_1, ..., \mathsf{decom}_n)$ denotes the sequence of decommitments generated by C in the decommit stage.

Hybrid H_b^ε: This hybrid is identical to H_b, except that the execution of OT_{i^*} in Step 3 of the commit stage is replaced with its ε-simulation.

$\underline{\mathsf{Out}_{H_b} \overset{\mathrm{c}}{\approx}_\varepsilon \mathsf{Out}_{H_b^\varepsilon}}$: This follows directly from ε-simulation security of OT against malicious receivers.

$\underline{\mathsf{Out}_{H_0^\varepsilon} \overset{\mathrm{s}}{\approx} \mathsf{Out}_{H_1^\varepsilon}}$: In H_b^ε, let $\{r^*_{i^*,j}\}_{j\in[k]}$ be the receiver's input to the ideal functionality of parallel OT provided by the simulator. Then the ideal functionality returns $\{s_{i^*,j}^{r_{i^*,j}}\}_{i^*\in[n],j\in[k]}$ to the simulator where $\{s^0_{i^*,j}, s^1_{i^*,j}\}_{j\in[k]}$ is the honest committer's input to OT_{i^*} generated according to the description of the protocol. Especially, no information of $\{s_{i^*,j}^{1-r_{i^*,j}}\}_{j\in[k]}$ is used until this point. Since $\{s^b_{i^*,j}\}_{j\in[k],b\in\{0,1\}}$ is a $2k$-out-of-$2k$ XOR secret sharing of m_{b,i^*}, which is the i^*-th component of $\mathbf{m}_b$, no information of m_{b,i^*} is revealed in H_b^ε unless all the remaining shares $\{s_{i^*,j}^{1-r_{i^*,j}}\}_{j\in[k]}$ are revealed at later stages, which happens only if $t_{i^*,j} = 1 - r_{i^*,j}$ for all $j \in [k]$ where $\{t_{i,j}\}_{i\in[n],j\in[k]}$ is the result of coin-flipping in Step 4 of the commit stage. However, by computational hiding of EqCom, the malicious receiver can cause only a negligible bias on the distribution of $\{t_{i,j}\}_{i\in[n],j\in[k]}$. Thus, the probability that $t_{i^*,j} = 1-r_{i^*,j}$ for all $j \in [k]$ is at most $2^{-k}+\mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$. Thus, with probability $1-\mathsf{negl}(\lambda)$, m_{b,i^*} remains information-theoretically hidden. This implies $\mathsf{Out}_{H_0^\varepsilon} \overset{\mathrm{s}}{\approx} \mathsf{Out}_{H_1^\varepsilon}$.

Combining the above, we obtain $\mathsf{Out}_{H_0} \overset{\mathrm{c}}{\approx}_{2\varepsilon} \mathsf{Out}_{H_1}$. Since this holds for any noticeable function ε, this implies $\mathsf{Out}_{H_0} \overset{\mathrm{c}}{\approx} \mathsf{Out}_{H_1}$, which means that the protocol satisfies computational hiding.

Extractability with Over-extraction. Let $\mathcal{SQ} = (\mathcal{SQ}_0, \mathcal{SQ}_1)$ be the simulation equivocator for EqCom. We construct the simulation extractor with over-extraction $\mathcal{SE}_{\mathrm{over}}$ as follows:
$\mathcal{SE}_{\mathrm{over}}^{C^*(\rho)}(1^\lambda)$:

1. Interact with C^* in Step 1 playing the role of R.
2. Execute n-parallel executions of OT in Step 3 with C^* where $\mathcal{SE}_{\mathrm{over}}$ plays the role of the honest receiver of OT that uses a independently and uniformly random bits $\{r_{i,j}\}_{j\in[k]}$ as input in OT_i for $i \in [n]$. Let $\{s^*_{i,j}\}_{j\in[k]}$ be the receiver's outcome of OT_i for $i \in [n]$.
3. Let ρ' be the internal state of C^* at the end of Step 3. Run $(\rho'', \mathsf{ST}_{\mathcal{SQ}}) \leftarrow \mathcal{SQ}_0^{C^*(\rho')}(1^\lambda)$. (Recall that C^* plays the role of receiver for EqCom.)

4. Resume C^* from Step 4b with its internal state ρ'' to obtain θ_C.
5. Let $\overline{r} \in \{0,1\}^{nk}$ be the string whose $(i-1)k+j$-th bit is $1-r_{i,j}$ for $i \in [n]$ and $j \in [k]$ and set $\theta_R := \overline{r} \oplus \theta_C$.
6. Run $\mathsf{EqCom.decom} \leftarrow \mathcal{SQ}_1(\mathsf{ST}_{\mathcal{SQ}}, \theta_R)$ and sends θ_R and $\mathsf{EqCom.decom}$ to C^*. Note that $t_{i,j}$ is now programmed to be $1-r_{i,j}$.
7. Receive $s_{i,j}^{t_{i,j}} = s_{i,j}^{1-r_{i,j}}$ from C^*. Note that it obtains all the shares $\{s_{i,j}^b\}_{i\in[n],j\in[k],b\in\{0,1\}}$ at this point.
8. Compute $m_{\mathsf{Ext},i} = \bigoplus_{j\in[k],b\in\{0,1\}} s_{i,j}^b$ for $i \in [n]$.
9. Output the transcript τ, the final state ST_{C^*} of C^*, and $\{m_{\mathsf{Ext},i}\}_{i\in[n]}$.

First, it is easy to see that

$$\Pr\left[\exists i \in [n] \ s.t. \ \mathsf{val}_i(\tau) \notin \{m_{\mathsf{Ext},i}, \bot\}\right] = 0$$

where $(\tau, \mathsf{ST}_{C^*}, \{m_{\mathsf{Ext},i}\}_{i\in[n]}) \leftarrow \mathcal{SE}_{\mathrm{over}}^{C^*(\rho)}(1^\lambda)$. To see this, suppose that $\mathsf{val}_i(\tau) \neq \bot$. In this case, there must exist secret sharing $\{s_{i,j}^b\}_{j\in[k],b\in\{0,1\}}$ of m_i that is consistent to the transcript. By the perfect correctness of OT, these shares are obtained by $\mathcal{SE}_{\mathrm{over}}$ and thus $m_{\mathsf{Ext},i} = m_i$.

Below, we prove

$$\left\{(\tau, \mathsf{ST}_{C^*}) : (\tau, \mathsf{ST}_{C^*}, \{m_{\mathsf{Ext},i}\}_{i\in[n]}) \leftarrow \mathcal{SE}_{\mathrm{over}}^{C^*(\rho)}(1^\lambda)\right\}_\lambda \stackrel{\mathrm{c}}{\approx} \left\{(\tau, \mathsf{ST}_{C^*}) : (\tau, \mathsf{ST}_{C^*}, b_{\mathrm{com}}) \leftarrow \langle C^*(\rho), R\rangle(1^\lambda)\right\}_\lambda. \tag{60}$$

We consider the following hybrids.

Hybrid H_0: This hybrid executes $(\tau, \mathsf{ST}_{C^*}, b_{\mathrm{com}}) \leftarrow \langle C^*(\rho), R\rangle(1^\lambda)$ and outputs $(\tau, \mathsf{ST}_{C^*})$.

Hybrid H_1: This hybrid is identical to the previous one except that the commitment by R of EqCom in Step 4a is generated by $\mathcal{SQ}_0$, which is decomitted to θ_R by $\mathcal{SQ}_1$ in Step 4c. Note that θ_R is just a uniformly random string that is independent of $\{r_{i,j}\}_{i\in[n],j\in[k]}$ in this hybrid.

$\underline{\mathsf{Out}_{H_0} \stackrel{\mathrm{c}}{\approx} \mathsf{Out}_{H_1}}$: This follows directly from equivocality of EqCom

Hybrid H_2: This hybrid is identical to the previous one except that R uses 0^k as input of OT_i for all $i \in [n]$ in Step 3.

$\underline{\mathsf{Out}_{H_1} \stackrel{\mathrm{c}}{\approx} \mathsf{Out}_{H_2}}$: This follows directly from indistinguishability-based security of OT against malicious senders.

Hybrid H_3: This hybrid is identical to the previous one except that θ_R is set as $\theta_R = \overline{r} \oplus \theta_C$ where $\overline{r}$ is as defined in the description of $\mathcal{SE}_{\mathrm{over}}$. Note that θ_R can depend on θ_C since θ_R is not used in Step 4a due to the modification made in H_1.

$\underline{\mathsf{Out}_{H_2} \stackrel{\mathrm{i.d.}}{=\!=} \mathsf{Out}_{H_3}}$: This follows directly from the observation that u_1 is a independently and uniformly random string in both hybrids noting that no information of r is used in Step 3 due to the modification made on H_2.

Hybrid H_4: This hybrid is identical to the previous one except that R uses $\{r_{i,j}\}_{j\in[k]}$ as input of OT_i for all $i \in [n]$ in Step 3.

$\underline{\mathsf{Out}_{H_3} \overset{c}{\approx} \mathsf{Out}_{H_4}}$: This follows directly from indistinguishability-based security of OT against malicious senders.

Now, we can see that H_4 just runs $(\tau, \mathsf{ST}_{C^*}) : (\tau, \mathsf{ST}_{C^*}, \{m_{\mathsf{Ext},i}\}_{i\in[n]}) \leftarrow \mathcal{SE}^{C^*(\rho)}_{\mathrm{over}}(1^\lambda)$ and outputs $(\tau, \mathsf{ST}_{C^*})$. Combining the above, we obtain Eq. (60). This completes the proof of extractability with over-extraction. □

14.3 Removing Over-extraction

Next, we give a compiler that upgrades extractable batch commitments with over-extraction (as per Def. 23) into one with full-fledged extractability without over-extraction (as per Def. 22). It is based on the cut-and-choose technique that is very similar to the one used for upgrading "weak" extractable commitments into "strong" one in [CCLY22a].[42] Our construction of an extractable batch commitment scheme with over-extraction is described in Prot. 8. It makes black-box use of the following building blocks:

1. The extractable batch commitment scheme $\mathsf{OverExtBCom}$ with over-extraction given in Prot. 7, which in turn makes black-box use of any OTs. Note that it is $(O(t(\lambda)) + \omega(1))$-round if the assumed OT is $t(\lambda)$-round (Thm. 17).
2. A commitment scheme ε-ExtCom that satisfies statistical binding, computational hiding, and extractability with ε-simulation (as per Def. 3). Constant-round and black-box construction of such a scheme based on OWFs is given in [CCLY22a].[43]
3. An $(n+1, k)$-perfectly verifiable secret sharing scheme $\mathsf{VSS} = (\mathsf{VSS}_{\mathsf{Share}}, \mathsf{VSS}_{\mathsf{Recon}})$ (as per Def. 8). We require that k is a constant fraction of n such that $k \le n/3$. There are known constructions (without any computational assumptions) satisfying these properties [BGW88, CDD+99].

Protocol 8: Extractable Batch Commitment $\mathsf{ExtBCom}$

Parameters. Let $n(\lambda)$ be a polynomial on λ. Let k be a constant fraction of n such that $k \le n/3$.

Input: Both the committer C and the receiver R get security parameter 1^λ as the common input. C in addition gets a sequence of strings $\mathbf{m} = (m_1, ..., m_N) \in \{0,1\}^{\ell(\lambda)\times N(\lambda)}$ as his private input, where $\ell(\cdot)$ and $N(\cdot)$ are polynomials.[a]

Commit Stage:

1. For $i \in [N]$, C prepares n views $\{\mathsf{v}_{i,j}\}_{j\in[n]}$, corresponding to an MitH execution for the $(n+1, k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the message m_i (see Rmk. 5 for details).
2. For $i \in [N]$, C and R involve an execution of $\mathsf{OverExtBCom}$ where C commits to $\mathbf{v} := \{\mathsf{v}_{i,j}\}_{i\in[N],j\in[n]} \in \{0,1\}^{\ell\times Nn}$.
3. C and R engage in the following coin-flipping subprotocol as detailed below.
   (a) R samples a random string θ_R of proper length and commits to it using ε-ExtCom.
   (b) C samples a random string θ_C of proper length and sends it to R.
   (c) R sends to C the value θ_R together with the corresponding decommitment information w.r.t. the ε-ExtCom in Step 3a. Now, C and R agree on a random string $\theta := \theta_R \oplus \theta_C$. By a proper choice of length, the string θ it can be interpreted as specifying N size-k random subsets of $[n]$. We write $(\eta_1, ..., \eta_N)$ to mean these subsets.

[42] We omit the definitions of weak and strong extractability in [CCLY22a] since this is not needed for our purpose.
[43] In fact, we only need a weaker security called "weak extractability with ε-simulation" in [CCLY22a].

4. For $i \in [N]$, C decommits to the VSS shares in the set η_i, i.e. it sends $\{\mathsf{v}_{i,j}\}_{j\in\eta_i}$ along with the corresponding decommitment information w.r.t. $\mathsf{OverExtBCom}$ in Step 2.
5. R checks the following conditions:
   (a) All the decommitments in Step 4 are valid; **and**
   (b) for any $i \in [N]$ and $j, j' \in \eta_i$, views $(\mathsf{v}_{i,j}, \mathsf{v}_{i,j'})$ are consistent (as per Def. 10 and Rmk. 3) w.r.t. the $\mathsf{VSS}_{\mathsf{Share}}$ execution in Step 1.

   If all the checks pass, R accepts (i.e., outputs $b_{\mathrm{com}} = \top$); otherwise, R rejects (i.e., outputs $b_{\mathrm{com}} = \bot$).

Decommit Stage:

1. For $i \in [N]$, C defines decom_i to be a string consisting of $\{\mathsf{v}_{i,j}\}_{j\in[n]}$ together with all the corresponding decommitment information w.r.t. $\mathsf{OverExtBCom}$ in Step 2 of the Commit Stage. C sends $\mathbf{m} = (m_1, ..., m_N)$ and $\mathsf{decom} = (\mathsf{decom}_1, ..., \mathsf{decom}_N)$.
2. For each $i \in [N]$, R runs the verification procedure Verify_i that works as follows:
   (a) Construct $\{\mathsf{v}'_{i,j}\}_{j\in[n]}$ as follows: in Step 1 of the Decommit Stage, if the decommitment to $\mathsf{v}_{i,j}$ is valid, R sets $\mathsf{v}'_{i,j} := \mathsf{v}_{i,j}$; otherwise, R sets $\mathsf{v}'_{i,j} := \bot$.
   (b) Accept if $m_i = \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{v}'_{i,1}, \ldots, \mathsf{v}'_{i,n})$ and otherwise reject.

   R accepts if Verify_i accepts for all $i \in [N]$ and otherwise rejects.

[a] We use N instead of n to mean the number of committed messages of $\mathsf{OverExtBCom}$ to avoid notational collision with the parameter for VSS.

Theorem 18. *Assuming the existence of $t(\lambda)$-round semi-honest OT protocols, there exists (i.e., Prot. 8) a black-box, $O(t(\lambda))+\omega(1)$-round construction of batch commitments that satisfies statistical binding, computational hiding (as per Def. 20), and extractability (as per Def. 22).*

Proof. Statistical binding property immediately follows from that of $\mathsf{OverExtBCom}$. Below, we show computational hiding and extractability.

Computational Hiding. Let $R^*(\rho)$ be any malicious QPT receiver. Fix $i^* \in [\ell]$ and pair of sequence of messages $\mathbf{m}_0$ and $\mathbf{m}_1$ that differ only on the i^*-th component. We consider the following hybrids for $b \in \{0, 1\}$ and noticeable function ε.

Hybrid H_b: This hybrid simulates execution between C with input $\mathbf{m}_b$ and $R^*(\rho)$ and outputs $(\mathsf{OUT}_{R^*}\langle C(\mathbf{m}_b), R^*\rangle(1^\lambda), \{\mathsf{decom}_i\}_{i\neq i^*})$ where $(\mathsf{decom}_1, ..., \mathsf{decom}_n)$ denotes the sequence of decommitments generated by C in the decommit stage.

Hybrid H_b^ε: This hybrid is identical to H_b, except for the following changes: It takes size-k random subsets $\eta_i \subseteq [n]$ for all $i \in [N]$ at the beginning. Then it runs the ε-simulation extractor for ε-ExtCom to extract θ_R while simulating the state of R^* in Step 3a and defines θ_C so that $\theta_R \oplus \theta_C$ specifies the subsets $(\eta_1, ..., \eta_N)$ in Step 3b.

$\mathsf{Out}_{H_b} \overset{\mathrm{c}}{\approx}_\varepsilon \mathsf{Out}_{H_b^\varepsilon}$: This follows directly from extractability with ε-simulation of ε-ExtCom noting that the distribution of θ_C is uniformly random in both hybrids.

$\mathsf{Out}_{H_0^\varepsilon} \overset{\mathrm{c}}{\approx} \mathsf{Out}_{H_1^\varepsilon}$: Note that the subset θ_{i^*} is fixed at the beginning in these hybrids. Then we can reduce computational indistinguishability of them to computational hiding of $\mathsf{OverExtBCom}$ by the same argument as the security proof of the VSS hiding game (Expr. 1).

Combining the above, we obtain $\mathsf{Out}_{H_0} \overset{c}{\approx}_{2\varepsilon} \mathsf{Out}_{H_1}$. Since this holds for any noticeable function ε, this implies $\mathsf{Out}_{H_0} \overset{c}{\approx} \mathsf{Out}_{H_1}$, which means that the protocol satisfies computational hiding.

Extractability. Let $\mathcal{SE}_{\mathrm{over}}$ be the simulation extractor with over-extraction for $\mathcal{SE}_{\mathrm{over}}$. We construct the simulation extractor $\mathcal{SE}$ as follows:

$\mathcal{SE}^{C^*(\rho)}(1^\lambda)$:

1. Run $(\mathsf{OverExtBCom}.\tau, \rho', \{\mathsf{v}_{\mathsf{Ext},i,j}\}_{i\in[N],j\in[n]}) \leftarrow \mathcal{SE}_{\mathrm{over}}^{C^*(\rho)}(1^\lambda)$ where $\mathsf{OverExtBCom}.\tau$ is the simulated transcript of the execution of $\mathsf{OverExtBCom}$ in Step 2, ρ' is the simulated state of C^* at the end of Step 2, and $\{\mathsf{v}_{\mathsf{Ext},i,j}\}_{i\in[N],j\in[n]}$ is the tuple of the extracted messages.
2. Run the rest of the commit stage while playing the role of the honest receiver R.
3. Let ST_{C^*} be the state of C^* at the end of the commit stage. Define $\{m_{\mathsf{Ext},i}\}_{i\in N}$ as follows:
   (a) If $b_{\mathrm{com}} = \bot$ (i.e., R rejects in Step 5 of the commit stage), then set $m_{\mathsf{Ext},i} := \bot$ for all $i \in [N]$.
   (b) Otherwise, set $m_{\mathsf{Ext},i} := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{v}_{\mathsf{Ext},i,1}, \dots, \mathsf{v}_{\mathsf{Ext},i,n})$ for $i \in [N]$.
4. Output $(\mathsf{ST}_{C^*}, \{m_{\mathsf{Ext},i}\}_{i\in[N]})$.

For $i \in [N]$, let Good_i be the event that there exists m_i^* such that $m_i^* = \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{v}'_{i,1}, \dots, \mathsf{v}'_{i,n})$ for all $\{\mathsf{v}'_{i,j}\}_{j\in[n]}$ such that $\mathsf{v}'_{i,j} = \mathsf{val}_{i,j}(\mathsf{OverExtBCom}.\tau)$ or $\mathsf{val}_{i,j}(\mathsf{OverExtBCom}.\tau) = \bot$ for all $j \in [n]$ where $\mathsf{OverExtBCom}.\tau$ is the transcript of $\mathsf{OverExtBCom}$ in Step 2 of the Commit Stage. Let Bad_i be the complementary event of Good_i. Then for any $i \in [N]$, we have

$$\Pr[\mathsf{Bad}_i \wedge b_{\mathrm{com}} = \top] = \mathsf{negl}(\lambda). \tag{61}$$

We omit its proof since almost identical claim is proven in [CCLY22a, Section 5.2].

It is easy to see that $m_{\mathsf{Ext},i} = m_i^* = \mathsf{val}_i(\tau)$ whenever Good_i occurs where τ is the transcript of the commit stage of $\mathsf{ExtBCom}$. By the union bound, Eq. (61) implies that Good_i occurs for all $i \in [N]$ whenever $b_{\mathrm{com}} = \top$ except for a negligible probability. Thus, whenever $b_{\mathrm{com}} = \top$, $m_{\mathsf{Ext},i} = \mathsf{val}_i(\tau)$ except for a negligible probability. Combined with extractability with over-extraction (as per Def. 23), this directly implies extractability (as per Def. 22). □

15 Black-Box Post-Quatnum ExtCom-and-Prove

15.1 Definition

The following definition is taken from [CCLY22a] with modifications to admit negl-simulation instead of ε-simulation for extractability and ZK.

Definition 24 (Simulatable ExtCom-and-Prove). *An ExtCom-and-Prove scheme consists of a pair of protocols $\Pi_{ECNP} = (\mathsf{ExtCom}, \mathsf{Prove})$ executed between a pair of* PPT *machines P and V. Let $m \in \{0,1\}^{\ell(\lambda)}$ (where $\ell(\cdot)$ is some polynomial) is a message that P wants to commit to. The protocol consists of the following stages (we omit the input 1^λ to P and V):*

– **Commit Stage:** *$P(m)$ and V execute ExtCom, which generates a transcript (commitment) com, P's state ST_P, and V's decision $b \in \{\top, \bot\}$ indicating acceptance (i.e., $b = \top$) or rejection (i.e., $b = \bot$). We denote this execution as $(\mathsf{com}, \mathsf{ST}_P, b) \leftarrow \langle P(m), V\rangle_{\mathsf{EC}}$. A malicious verifier is allowed to output any quantum state, which we denote by ST_{V^*} instead of b, and to keep the state for the prove stage.*

– **Decommit Stage:**[44] *$P(\mathsf{ST}_P)$ generates a decommitment* decom *and sends it to V along with a message m. V accepts or rejects.*

– **Prove Stage:** *Let ϕ be any predicate. $P(\mathsf{ST}_P, \phi)$ and $V(\mathsf{com}, \phi)$ execute* Prove*, after which V outputs $\top$ (accept) or $\bot$ (reject). We denote the execution of this stage as $b' \leftarrow \langle P(\mathsf{ST}_P), V(\mathsf{com})\rangle^{\phi}_{\mathsf{Pr}}$, where $b' \in \{\top, \bot\}$ is V's output. A malicious verifier is allowed to output an arbitrary quantum state, which we denote by OUT_{V^*} instead of b'.*

The following requirements are satisfied:

1. **Security as Simulation Extractable Commitment.** *The Commit Stage and Decommit Stage constitute a post-quantum commitment scheme (as per Def. 1 where P and V play the roles of C and R, respectively) that is computationally hiding, statistically binding, and extractable. Here, the extractability means the following: There exists a QPT algorithm $\mathcal{SE}$ (called the simulation extractor) such that for any non-uniform QPT $C^*(\rho)$,*

$$\left\{\mathcal{SE}^{C^*(\rho)}(1^\lambda)\right\}_\lambda \stackrel{c}{\approx} \left\{(\mathsf{val}(\tau), \mathsf{ST}_{C^*}) : (\tau, \mathsf{ST}_{C^*}, b_{\mathrm{com}}) \leftarrow \langle C^*(\rho), R\rangle(1^\lambda)\right\}_\lambda,$$

where $\mathsf{val}(\tau)$ is the value committed by C^* as defined in Def. 2.[45]

2. **Completeness.** *For any $m \in \{0,1\}^{\ell(\lambda)}$ and any polynomial-time computable predicate ϕ s.t. $\phi(m) = 1$, it holds that*

$$\Pr\left[b = \top \ \wedge \ b' = \top : \begin{array}{l}(\mathsf{com}, \mathsf{ST}_P, b) \leftarrow \langle P(m), V\rangle_{\mathsf{EC}} \\ b' \leftarrow \langle P(\mathsf{ST}_P), V(\mathsf{com})\rangle^{\phi}_{\mathsf{Pr}}\end{array}\right] = 1. \tag{62}$$

3. **Soundness.** *For any predicate ϕ and any non-uniform QPT prover $P^*(\rho)$,*

$$\Pr\left[\begin{array}{l}b = \top \ \wedge \ b' = \top \\ \wedge \ \phi(\mathsf{val}_{\mathsf{ExtCom}}(\mathsf{com})) = 0\end{array} : \begin{array}{l}(\mathsf{com}, \mathsf{ST}_{P^*}, b) \leftarrow \langle P^*(\rho), V\rangle_{\mathsf{EC}} \\ b' \leftarrow \langle P^*(\mathsf{ST}_{P^*}), V(\mathsf{com})\rangle^{\phi}_{\mathsf{Pr}}\end{array}\right] = \mathsf{negl}(\lambda), \tag{63}$$

where $\mathsf{val}_{\mathsf{ExtCom}}(\mathsf{com})$ is as defined in Def. 2 and we stipulate that $\phi(\bot) = 0$.

4. **Zero-Knowledge.** *There exists a pair of QPT simulators $(\mathcal{S}_{\mathsf{EC}}, \mathcal{S}_{\mathsf{Pr}})$ such that for any $m \in \{0,1\}^{\ell(\lambda)}$, polynomial-time computable predicate ϕ s.t. $\phi(m) = 1$, any non-uniform QPT verifier $V^*(\rho)$, and any noticeable function $\varepsilon(\lambda)$, the following conditions hold:*

$$\left\{\widetilde{\mathsf{ST}}_{V^*} : (\widetilde{\mathsf{ST}}_{V^*}, \mathsf{ST}_{\mathsf{EC}}) \leftarrow \mathcal{S}^{V^*(\rho)}_{\mathsf{EC}}\right\}_\lambda \stackrel{c}{\approx} \left\{\mathsf{ST}_{V^*} : (\mathsf{com}, \mathsf{ST}_P, \mathsf{ST}_{V^*}) \leftarrow \langle P(m), V^*(\rho)\rangle_{\mathsf{EC}}\right\}_\lambda \tag{64}$$

$$\left\{\begin{array}{l}\widetilde{\mathsf{OUT}}_{V^*} : (\widetilde{\mathsf{ST}}_{V^*}, \mathsf{ST}_{\mathsf{EC}}) \leftarrow \mathcal{S}^{V^*(\rho)}_{\mathsf{EC}} \\ \widetilde{\mathsf{OUT}}_{V^*} \leftarrow \mathcal{S}^{V^*}_{\mathsf{Pr}}(1^{\varepsilon^{-1}}, \widetilde{\mathsf{ST}}_{V^*}, \mathsf{ST}_{\mathsf{EC}}, \phi)\end{array}\right\}_\lambda \stackrel{c}{\approx} \left\{\mathsf{OUT}_{V^*} : \begin{array}{l}(\mathsf{com}, \mathsf{ST}_P, \mathsf{ST}_{V^*}) \leftarrow \langle P(m), V^*(\rho)\rangle_{\mathsf{EC}} \\ \mathsf{OUT}_{V^*} \leftarrow \langle P(\mathsf{ST}_P), V^*(\mathsf{ST}_{V^*})\rangle^{\phi}_{\mathsf{Pr}}\end{array}\right\}_\lambda. \tag{65}$$

We refer to $\mathcal{S}_{\mathsf{EC}}$ (resp. $\mathcal{S}_{\mathsf{Pr}}$) as the Commit-Stage (resp. Prove-Stage) simulator.

15.2 Construction of ExtCom-and-Prove

We construct an ExtCom-and-Prove scheme based on extractable batch commitments. The construction is almost identical to that in [CCLY22a, Section 6.5] except that we require full-simulation security instead of ε-simulation.

The construction is shown in Prot. 9. It makes black-box use of the following building blocks:

[44] This stage is rarely executed in applications.

[45] We only require computational indistinguishability rather than statistical one unlike Def. 3. This is inherited from Def. 22 (see also Rmk. 11).

1. The extractable batch commitment (as per Def. 22) $\mathsf{ExtBCom}$ (Prot. 8), which in turn makes black-box use of any OTs. Note that it is $(O(t(\lambda))+\omega(1))$-round if the assumed OT is $t(\lambda)$-round (Thm. 18).

2. An equivocal commitment scheme EqCom (as per Def. 19). A $\omega(1)$-round construction of it is known assuming only black-box access to post-quantum secure OWFs (Thm. 14).

3. A constant-round statistically-binding, computationally-hiding commitment Com, (e.g., Naor's commitment).

4. An $(n+1,k)$-perfectly verifiable secret sharing scheme $\mathsf{VSS} = (\mathsf{VSS}_{\mathsf{Share}}, \mathsf{VSS}_{\mathsf{Recon}})$ (as per Def. 8). We require that k is a constant fraction of n such that $k \leq n/3$. There are known constructions (without any computational assumptions) satisfying these properties [BGW88, CDD+99].

5. A (n,k)-perfectly secure MPC protocol Π_{MPC} (as per Def. 9).

Protocol 9: ExtCom-and-Prove scheme $\Pi_{\mathbf{ECnP}}$

Parameter Setting: Let $n(\lambda)$ be a polynomial on λ. Let k be a constant fraction of n such that $k \leq n/3$.

Input: Both the prover P and the verifier V get 1^λ as the common input. P in addition gets a string $m \in \{0,1\}^{\ell(\lambda)}$ as his private input, where $\ell(\cdot)$ is a polynomial.

Commit Stage:

1. P prepares n views $\{\mathsf{v}_i\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string x (see Rmk. 5 for details).
2. P and V involve in $\mathsf{ExtBCom}$, where P commits to $(\mathsf{v}_1, ..., \mathsf{v}_n)$.

Decommit Stage:

1. P sends m and $\{\mathsf{v}_i\}_{i\in[n]}$ together with the corresponding decommitment information w.r.t. the $\mathsf{ExtBCom}$ in Step 2 of the Commit Stage.
2. V checks that all the decommitments in Step 1 of the Decommit Stage are valid and $m = \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{v}_1, \ldots, \mathsf{v}_n)$. If so, it accepts and otherwise rejects.

Prove Stage: Both parties learn a polynomial-time computable predicate ϕ.

1. P prepares n views $\{\mathsf{v}'_i\}_{i\in[n]}$ corresponding to an (n,k)-MitH execution for the functionality F_ϕ described below, where party P_i uses v_i as input. It then commits to each of these views v'_i independently in parallel using Com.
   - **Functionality F_ϕ:** This collects inputs v_i from party i, runs $\mathsf{VSS}_{\mathsf{Recon}}$ on these inputs to recover a value x, and outputs $\phi(x)$.
2. P and V engage in the following coin-flipping subprotocol as detailed below.
   (a) P samples a random string θ_P of proper length and commits to it using EqCom.
   (b) V samples a random string θ_V of proper length and sends it to P.
   (c) P sends to V the value θ_P together with the corresponding decommitment information w.r.t. the EqCom in Step 2a. Now, P and V agree on a random string $\theta := \theta_P \oplus \theta_V$. By a proper choice of length, the string θ it can be interpreted as specifying a size-k random subset $\eta \subset [n]$.
3. P sends to V in *one round* the following messages:

(a) $\{\mathsf{v}_i\}_{i\in\eta}$ together with the corresponding decommitment information w.r.t. the $\mathsf{ExtBCom}$ in Step 2 of the Commit Stage; **and**
(b) $\{\mathsf{v}'_i\}_{i\in\eta}$ together with the corresponding decommitment information w.r.t. the Com in Step 1 of the Prove Stage.

4. V checks the following conditions:
(a) All the decommitments in Steps 3a and 3b are valid; **and**
(b) for any $i \in \eta$, v_i is the prefix of v'_i ; **and**
(c) for any $i, j \in \eta$, views $(\mathsf{v}'_i, \mathsf{v}'_j)$ are consistent (as per Def. 10 and Rmk. 3) w.r.t. the $\mathsf{VSS}_{\mathsf{Share}}$ execution in Step 1 of the Commit Stage and the Π_{MPC} execution as described in Step 1 of the Prove Stage.

If all the checks pass, V accepts; otherwise, V rejects.

Theorem 19. *Assume the existence of $t(\lambda)$-round semi-honest OTs. Then, there exists a $(O(t(\lambda))+\omega(1)$-round construction of an ExtCom-and-Prove scheme Π_{ECNP} (i.e., Prot. 9) satisfying Def. 24. Moreover, this construction makes only black-box use of the assumed OT.*

This can be proven similarly to the security proof of ε-ExtCom-and-Prove in [CCLY22a, Section 6.5]. The only differences from their construction are that

1. we use fully-simulatable extractable batch commitments to commit to the views in Step 2 whereas they use ε-simulatable parallel extractable commitments, and
2. we implement the coin-flipping subprotocol Step 2 using equivocal commitments instead whereas they used ε-simulation extractable commitments.

The difference between full-simulation and ε-simulation is directly connected to that we achive full-simulation instead of ε-simulation for the resulting protocol. For the coin-flipping subprotocol, what we need here is *one-sided* simulation security where we require simulation-based security against malicious verifiers but only require a weaker security against malicious provers that they cannot bias the result of coin-flipping. This can be achieved using either equivocal commitments as is done here or extractable commitments as is done in [CCLY22a, Section 6.5]. With the above remarks in mind, it is straightforward to adapt the proof in [CCLY22a, Section 6.5] to our setting. Thus, we omit the proof of Thm. 19.

16 Post-Quantum Black-Box MPC with Full Simulation

16.1 Black-Box PQ-2PC with Full Simulation

In this part, we prove the following theorem.

Theorem 20. *Assuming the existence of a constant-round, semi-honest post-quantum OT, there exists a black-box, $\omega(1)$-round construction of post-quantum 2PC.*

To prove Thm. 20, we follow the paradigm established in earlier works, in particular [GLSV21, CCLY22a]. This involves two steps.

Step-1: In [CCLY22a], the authors first define an ideal functionality $\mathcal{F}^t_{\text{SO-COM}}$ for "selective-opening secure" commitments, which is shown in Fig. 2. More descriptively, this is an idealization of a commitment that offers *selective opening* security in a *bounded-parallel* execution. That is, it can be used by a committer to commit to an a-priori bounded number, say a polynomial $t(\lambda)$, of strings within a single invocation; later, the receiver may specify an arbitrary subset $I \subset [t]$ of positions, and the committer must decommit to the i-th commitment it made, for each $i \in I$.

The intuitive benefit in having access to such a construct arises from the fact that it allows for implementations of *cut-and-choose* protocols which naturally involve committing to several instances of certain data and then later opening a receiver-chosen subset of these committed instances. The techniques we use to get 2PC will involve these techniques.

Figure 2: The Ideal Functionality $\mathcal{F}^t_{\text{SO-COM}}$ [GLSV21, CCLY22a]

Commit Stage: $\mathcal{F}^t_{\text{SO-COM}}$ receives from the committer C a query $\big(\mathsf{Commit}, sid, (m_1, \ldots, m_t)\big)$. $\mathcal{F}^t_{\text{SO-COM}}$ records $\big(sid, (m_1, \ldots, m_t)\big)$ and sends $(\mathsf{Receipt}, sid)$ to the receiver R. $\mathcal{F}^t_{\text{SO-COM}}$ ignores further Commit messages with the same sid.

Decommit Stage: $\mathcal{F}^t_{\text{SO-COM}}$ receives from R a query $(\mathsf{Reveal}, sid, I)$, where I is a subset of $[t]$. If no $\big(sid, (m_1, \ldots, m_t)\big)$ has been recorded, $\mathcal{F}^t_{\text{SO-COM}}$ does nothing; otherwise, it sends to R the message $\big(\mathsf{Open}, sid, \{m_i\}_{i\in I}\big)$.

Step 2: Then, it is shown in [CCLY22a, Section 7.4] that $\mathcal{F}^t_{\text{SO-COM}}$ is indeed black-box 2PC-complete. Namely, [CCLY22a, Section 7.4] shows that given a protocol π that securely implements $\mathcal{F}^t_{\text{SO-COM}}$ against QPT adversaries, one can construct a *general-purpose* 2PC protocol (i.e, computing any efficient 2-party functionality) that is secure against QPT adversaries. Moreover, the 2PC construction makes only *black-box* use of π and involves only a constant multiplicative blow up in the number of rounds (as compared to π).

We must keep in mind the following caveat: the protocol π as described in [CCLY22a, Section 7.4] in fact implements $\mathcal{F}^t_{\text{SO-COM}}$ w.r.t. ε-simulation. Hence the final 2PC they obtained is also w.r.t. ε-simulation. It is straightforward however to see that the same proof works w.r.t. standard negligibly-close simulation as well. Namely, if one starts with a π that implements $\mathcal{F}^t_{\text{SO-COM}}$ w.r.t. the standard notion of negligible-close simulation, then the resulting 2PC protocol will also be secure w.r.t. the standard notion of negligible-close simulation.

Implementing $\mathcal{F}^t_{\text{SO-COM}}$. From the above discussion we can see that to prove Thm. 20, it suffices to construct a $\omega(1)$-round, black-box, post-quantum protocol implementing the $\mathcal{F}^t_{\text{SO-COM}}$ functionality. For that, we will make use of the $\omega(1)$-round ExtCom-and-Prove protocol described in Prot. 8. Note that it is okay to make use of this ExtCom-and-Prove protocol because this protocol makes only black-box use of a semi-honest post-quantum OT protocol, which is indeed the minimal assumption for our current goal of 2PC.

We can then conclude the proof of Thm. 20 using the following lemma.

Lemma 30 ([CCLY22a, Lemma 26]). *Assume the existence of Post-Quantum ExtCom-and-Prove (as per Def. 24). Then, for any polynomial $t(\lambda)$, there exists a post-quantum protocol implementing $\mathcal{F}^t_{\text{SO-COM}}$. Moreover, this construction makes only black-box use of the ExtCom-and-Prove protocol and incurs only a constant blow up in the number of rounds.*

Proof. This proof is essentially identical to the proof of [CCLY22a, Lemma 26], relying on the extractable commit-and-prove protocol given in Prot. 9. The idea is simple: the committer uses the Commit Stage of the Extcom-and-Prove protocol to commit to different messages $(m_1, \ldots, m_t)$ of its choice. Next, when required to decommit to a certain subset $I \subset [t]$ of messages, the committer reveals these messages $\{m_i\}_{i\in I}$ to the receiver and then uses to Prove Stage to prove that the revealed messages are indeed the committed ones for the appropriate positions. Note that this protocol then is purely black-box and only adds a small constant number of rounds of communication

(relating to sending the revealed subset and the decommitment information) over the underlying Extcom-and-Prove protocol.

Security against a cheating committer is obtained via the soundness of Prot. 9, and that against a cheating reciever can be seen from the zero-knowledge of the same underlying protocol. The crucial (and in fact only) difference is that [CCLY22a, Lemma 26] uses an underlying ExtCom-and-Prove protocol that offers ε-simulation. This is why they only manage to obtain a protocol implementing $\mathcal{F}^t_{\text{SO-COM}}$ w.r.t. ε-simulation. In contrast, Prot. 9 does achieve the standard notion of negligibly close simulation. It is then easy to verify that our protocol for $\mathcal{F}^t_{\text{SO-COM}}$ achieves the standard notion of full simulation as well, using the same proof. □

16.2 Black-Box PQ-MPC with Full Simulation

Here we turn to the problem of obtaining a *fully simulatable* black-box post-quantum MPC protocol. More precisely, we show the following theorem.

Theorem 21. *Assuming the existence of a semi-honest post-quantum OT, there exists a black-box construction of post-quantum MPC in polynomial rounds.*

This theorem follows directly from Thm. 20 and [IPS08]. To start, we observe that Thm. 20 provides a black-box construction of post-quantum maliciously secure and fully simulatable OT — via the constructed 2PC protocol (recall that the latter can be made to implement *any* 2 party functionality). We have further seen in Sec. 12 that the black-box compiler given in [IPS08] from OT to MPC works in the post-quantum setting as is.

There is however a caveat: recall that the original [IPS08] result is in the OT hybrid model in the UC setting, where the OT primitive is modeled as an ideal UC functionality. Such modeling indeed allows parallel OT calls (this has been observed and discussed in [CCLY22a, Section 7] and in Sec. 12). But the OT protocol we obtain from Thm. 20 is only secure in the *standalone* setting.

Fortunately, this does not become a problem for our application. Recall that the IPS compiler involves carrying out a certain polynomial number of OT calls or executions at the start of the protocol, which are carried out in parallel as observed above. For our purposes, we simply make required number of OT calls in sequence instead of in parallel, which adds to the round complexity of our protocol but preserves the desired order asymptotics — it is easy to check that our overall MPC protocol still takes only polynomial rounds in n (and thus also in λ).

References

Aar05. Scott Aaronson. Limitations of quantum advice and one-way communication. *Theory Comput.*, 1(1):1–28, 2005. 72

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. 1, 5

BCKM21. 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. 1, 3, 20, 91, 92, 95

BGW88. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, IL, USA, May 2–4, 1988. ACM Press. 28, 29, 101, 105

BKP19. Nir Bitansky, Dakshita Khurana, and Omer Paneth. Weak zero-knowledge beyond the black-box barrier. In Moses Charikar and Edith Cohen, editors, *51st Annual ACM Symposium on Theory of Computing*, pages 1091–1102, Phoenix, AZ, USA, June 23–26, 2019. ACM Press. 2

BLS22. Nir Bitansky, Huijia Lin, and Omri Shmueli. Non-malleable commitments against quantum attacks. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 519–550, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany. 26

BQSY24. John Bostanci, Luowen Qian, Nicholas Spooner, and Henry Yuen. An efficient quantum parallel repetition theorem and applications, 2024. Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024. 5

BS20. Nir Bitansky and Omri Shmueli. Post-quantum zero knowledge in constant rounds. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd Annual ACM Symposium on Theory of Computing*, pages 269–279, Chicago, IL, USA, June 22–26, 2020. ACM Press. 33

CCLL24. Nai-Hui Chia, Kai-Min Chung, Xiao Liang, and Jiahui Liu. The black-box simulation barrier persists in a fully quantum world. *CoRR*, abs/2409.06317, 2024. 1, 8

CCLY22a. Nai-Hui Chia, Kai-Min Chung, Xiao Liang, and Takashi Yamakawa. Post-quantum simulatable extraction with minimal assumptions: Black-box and constant-round. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 533–563, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. 2, 7, 8, 11, 18, 19, 20, 23, 24, 25, 28, 31, 32, 33, 37, 45, 55, 91, 97, 101, 103, 104, 106, 107, 108

CCLY22b. Nai-Hui Chia, Kai-Min Chung, Qipeng Liu, and Takashi Yamakawa. On the impossibility of post-quantum black-box zero-knowledge in constant round. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 59–67. IEEE, 2022. 1, 2, 8, 24

CCY21. Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. A black-box approach to post-quantum zero-knowledge in constant rounds. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 315–345, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. 28, 33

CDD+99. Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 28, 58, 101, 105

CDMW09. Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 387–402. Springer, Heidelberg, Germany, March 15–17, 2009. 1, 6, 82

CGMA85. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press. 28

CK88. Claude Crépeau and Joe Kilian. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *29th Annual Symposium on Foundations of Computer Science*, pages 42–52, White Plains, NY, USA, October 24–26, 1988. IEEE Computer Society Press. 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. 6

CMSZ21. Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments: Breaking the quantum rewinding barrier. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 49–58. IEEE, 2021. 16, 17, 18, 67, 70

DDN91. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press. 4

DNRS99. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 523–534. IEEE Computer Society, 1999. 2

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. 2

GKM+00. Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *41st Annual Symposium on Foundations of Computer Science*, pages 325–335, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. 3

GLM+04. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 258–277, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 1

GLM23. Vipul Goyal, Xiao Liang, and Giulio Malavolta. On concurrent multi-party quantum computation. In *Annual International Cryptology Conference*, pages 129–161. Springer, 2023. 1, 5

GLOV12. Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd Annual Symposium on Foundations of Computer Science*, pages 51–60, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press. 4, 13, 30, 31, 37

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. 1, 3, 106, 107

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. 1, 6

Gol04. Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004. 27

Goy11. Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 695–704, San Jose, CA, USA, June 6–8, 2011. ACM Press. 1, 4, 6, 7, 8, 85, 89, 90

GW11. Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press. 15

Hai08. Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 412–426, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany. 1, 6

Hof11. Dennis Hofheinz. Possibility and impossibility results for selective decommitments. *Journal of Cryptology*, 24(3):470–516, July 2011. 19

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. 5, 6

IKLP06. Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In Jon M. Kleinberg, editor, *38th Annual ACM Symposium on Theory of Computing*, pages 99–108, Seattle, WA, USA, May 21–23, 2006. ACM Press. 1, 6, 88

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th Annual ACM Symposium on Theory of Computing*, pages 21–30, San Diego, CA, USA, June 11–13, 2007. ACM Press. 1, 13, 30, 32, 33

IL89. Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, Research Triangle Park, NC, USA, October 30 – November 1, 1989. IEEE Computer Society Press. 3

IPS08. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 1, 3, 4, 6, 7, 19, 89, 90, 108

IR89. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, WA, USA, May 15–17, 1989. ACM Press. 1

JKKR17. Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 158–189, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany. 2

Kil88. Joe Kilian. Founding cryptography on oblivious transfer. In *20th Annual ACM Symposium on Theory of Computing*, pages 20–31, Chicago, IL, USA, May 2–4, 1988. ACM Press. 20, 91

Kil94. Joe Kilian. On the complexity of bounded-interaction and noninteractive zero-knowledge proofs. In *35th Annual Symposium on Foundations of Computer Science*, pages 466–477, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press. 20, 91

KQT25. William Kretschmer, Luowen Qian, and Avishay Tal. Quantum-computable one-way functions without one-way functions. In *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC)*, 2025. 3

Lin16. Yehuda Lindell. How to simulate it - A tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. https://eprint.iacr.org/2016/046. 27

LMS22. Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited or: How to do quantum rewinding undetectably. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 851–859. IEEE, 2022. 2

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. 5, 6

LPY23a. Xiao Liang, Omkant Pandey, and Takashi Yamakawa. A new approach to post-quantum non-malleability. In *2023 IEEE 64nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2023. ii, 1, 4, 5, 8, 10, 11, 12, 13, 14, 15, 25, 41, 42, 63, 66, 82

LPY23b. Xiao Liang, Omkant Pandey, and Takashi Yamakawa. A new approach to post-quantum non-malleability. 2023. https://arxiv.org/abs/2207.05861v3. 1, 42, 63

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. 2

Nao03. Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. 15

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. 4

PR05. Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 533–542, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 15, 65, 113

PW09. Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 403–418. Springer, Heidelberg, Germany, March 15–17, 2009. 1, 4, 6, 20, 91

RTV04. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 1

Wat09. John Watrous. Zero-knowledge against quantum attacks. *SIAM Journal on Computing*, 39(1):25–58, 2009. 33, 92

Wee10. Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st Annual Symposium on Foundations of Computer Science*, pages 531–540, Las Vegas, NV, USA, October 23–26, 2010. IEEE Computer Society Press. 1, 4, 6, 7, 8, 82, 85, 88, 89, 90

Wyn75. Aaron D Wyner. The wire-tap channel. *Bell system technical journal*, 54(8):1355–1387, 1975. 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

Zha20. Mark Zhandry. Schrödinger's pirate: How to trace a quantum decoder. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 61–91, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany. 16, 18, 67, 68

A Full Description of the Two-Sided PQ-MNC Protocol

In this section, we present the full description of the 1-1 PQ-NMC protocol without the 'one-sided' restriction. As explained in Sec. 8, this is obtained by applying the [PR05] 'two-slot' technique to Prot. 3.

The protocol is presented in Prot. 10. It makes use of exactly the same building blocks for Prot. 3 (as listed at the beginning of Sec. 5.1).

Protocol 10: (Two-Sided) PQ-NMC: Black-Box and Constant-Round

Parameter Setting: The tag space is defined to be $[T]$, where T is a polynomial in the security parameter λ. Let n be a polynomial in λ, and k be a constant fraction of n such that $2k \leq n/3$.

Input: Both the committer C and the receiver R get the security parameter 1^λ and a tag $t \in [T]$ as the common input; C gets a string $m \in \{0,1\}^{\ell(\lambda)}$ as its private input, where $\ell(\cdot)$ is a polynomial.

Commit Phase:

1. **(Initial Com to m.)** In this stage, C commits to the message with MitH.
   - C prepares n views $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,2k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the message m. C commits to each $\mathsf{cv}_i^{(1)}$ $(i \in [n])$ independently in parallel, using Naor's commitment.
2. **(Hard-Puzzle-A.)** In this stage, R sets up a t-solution hard puzzle. It then commits to one solution of the puzzle and proves in zero-knowledge the consistency with MitH. This corresponds to the **Slot-A** as described in Sec. 8.
   (a) C samples a size-k random subset $\mathsf{ch}_A \subseteq [n]$, and commits to it using ExtCom.
   (b) R samples t random strings $x_1^A, \ldots, x_t^A \xleftarrow{\$} \{0,1\}^\lambda$. R prepares n views $\{\mathsf{rv}_i^{(1,A)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string $x_1^A \| \ldots \| x_t^A$. R commits to each $\mathsf{rv}_i^{(1,A)}$ $(i \in [n])$ independently in parallel, using Naor's commitment.
   (c) R prepares another n views $\{\mathsf{rv}_i^{(2,A)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string $1\|x_1^A$. R commits to each $\mathsf{rv}_i^{(2,A)}$ $(i \in [n])$ independently in parallel, using ExtCom.
   (d) R then prepares another n views $\{\mathsf{rv}_i^{(3,A)}\}_{i\in[n]}$, corresponding to an (n,k)-MitH execution of the n-party functionality $F_{\mathsf{consis}}^{R,A}$ described below, where party P_i $(i \in [n])$ uses $\mathsf{rv}_i^{(1,A)} \| \mathsf{rv}_i^{(2,A)}$ as input. R commits to each $\mathsf{rv}_i^{(3,A)}$ $(i \in [n])$ independently in parallel, using Naor's commitment.
      - **Functionality $F_{\mathsf{consis}}^{R,A}$:** It collects input (and parses it as) $\mathsf{rv}_i^{(1,A)} \| \mathsf{rv}_i^{(2,A)}$ from party i for each $i \in [n]$. It then runs the recovery algorithm of VSS to obtain $a_1 \| \ldots \| a_t := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{rv}_1^{(1,A)}, \ldots, \mathsf{rv}_n^{(1,A)})$ and $j \| b_j := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{rv}_1^{(2,A)}, \ldots, \mathsf{rv}_n^{(2,A)})$. If $j \in [t]$ and $b_j = a_j$, it outputs 1 to each party; otherwise, it outputs 0 to each party.
   (e) C sends ch_A together with the decommitment information (w.r.t. Step 2a).
   (f) R sends $\{(\mathsf{rv}_i^{(1,A)}, \mathsf{rv}_i^{(2,A)}, \mathsf{rv}_i^{(3,A)})\}_{i\in\mathsf{ch}_A}$ together with the decommitment information (w.r.t. their respective commitments in Steps 2b to 2d).
   (g) C checks the validity of the decommitment information and the consistency among the revealed views $\{(\mathsf{rv}_i^{(1,A)}, \mathsf{rv}_i^{(2,A)}, \mathsf{rv}_i^{(3,A)})\}_{i\in\mathsf{ch}_A}$. In particular, it checks for each $i \in \mathsf{ch}_A$ that

$\mathsf{rv}_i^{(1,A)} \| \mathsf{rv}_i^{(2,A)}$ is the prefix of $\mathsf{rv}_i^{(3,A)}$. It also checks for each distinct pair $i, j \in \mathsf{ch}_A$ that $(\mathsf{rv}_i^{(1,A)}, \mathsf{rv}_j^{(1,A)})$ are consistent, $(\mathsf{rv}_i^{(2,A)}, \mathsf{rv}_j^{(2,A)})$ are consistent, and $(\mathsf{rv}_i^{(3,A)}, \mathsf{rv}_j^{(3,A)})$ are consistent, where by 'consistent' we refer to the consistency requirements as per Def. 10 and Rmk. 3. It also checks for each $i \in \mathsf{ch}_A$ the final output of P_i contained in $\mathsf{rv}_i^{(3,A)}$ is 1. It aborts immediately if any of the checks fail.

3. **(Hard-Puzzle-B.)** In this stage, R sets up a $(T-t)$-solution hard puzzle. It then commits to one solution of the puzzle and proves in zero-knowledge the consistency with MitH. This corresponds to the **Slot-B** as described in Sec. 8.

   (a) C samples a size-k random subset $\mathsf{ch}_B \subseteq [n]$, and commits to it using ExtCom.

   (b) R samples $(T-t)$ random strings $x_1^B, \ldots, x_{T-t}^B \xleftarrow{\$} \{0,1\}^\lambda$. R prepares n views $\{\mathsf{rv}_i^{(1,B)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string $x_1^B \| \ldots \| x_t^B$. R commits to each $\mathsf{rv}_i^{(1,B)}$ $(i \in [n])$ independently in parallel, using Naor's commitment.

   (c) R prepares another n views $\{\mathsf{rv}_i^{(2,B)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the string $1 \| x_1^B$. R commits to each $\mathsf{rv}_i^{(2,B)}$ $(i \in [n])$ independently in parallel, using ExtCom.

   (d) R then prepares another n views $\{\mathsf{rv}_i^{(3,B)}\}_{i\in[n]}$, corresponding to an (n,k)-MitH execution of the n-party functionality $F_{\mathsf{consis}}^{R,B}$ described below, where party P_i $(i \in [n])$ uses $\mathsf{rv}_i^{(1,B)} \| \mathsf{rv}_i^{(2,B)}$ as input. R commits to each $\mathsf{rv}_i^{(3,B)}$ $(i \in [n])$ independently in parallel, using Naor's commitment.

   – **Functionality $F_{\mathsf{consis}}^{R,B}$:** It collects input (and parses it as) $\mathsf{rv}_i^{(1,B)} \| \mathsf{rv}_i^{(2,B)}$ from party i for each $i \in [n]$. It then runs the recovery algorithm of VSS to obtain $a_1 \| \ldots \| a_{T-t} := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{rv}_1^{(1,B)}, \ldots, \mathsf{rv}_n^{(1,B)})$ and $j \| b_j := \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{rv}_1^{(2,B)}, \ldots, \mathsf{rv}_n^{(2,B)})$. If $j \in [T-t]$ and $b_j = a_j$, it outputs 1 to each party; otherwise, it outputs 0 to each party.

   (e) C sends ch_B together with the decommitment information (w.r.t. Step 3a).

   (f) R sends $\{(\mathsf{rv}_i^{(1,B)}, \mathsf{rv}_i^{(2,B)}, \mathsf{rv}_i^{(3,B)})\}_{i\in\mathsf{ch}_B}$ together with the decommitment information (w.r.t. their respective commitments in Steps 3b to 3d).

   (g) C checks the validity of the decommitment information and the consistency among the revealed views $\{(\mathsf{rv}_i^{(1,B)}, \mathsf{rv}_i^{(2,B)}, \mathsf{rv}_i^{(3,B)})\}_{i\in\mathsf{ch}_B}$ in the same manner as in Step 2g. It also checks for each $i \in \mathsf{ch}_B$ the final output of P_i contained in $\mathsf{rv}_i^{(3,B)}$ is 1. It aborts immediately if any of the checks fail.

4. **(ExtCom to m.)** C commits to m once again with an extractable MitH.

   – C prepares n views $\{\mathsf{cv}_i^{(2)}\}_{i\in[n]}$, corresponding to an MitH execution for the $(n+1,2k)$-$\mathsf{VSS}_{\mathsf{Share}}$ of the message m. C commits to each $\mathsf{cv}_i^{(2)}$ $(i \in [n])$ independently in parallel, using ExtCom.

5. **(Puzzle Solution Reveal.)** R reveals $(x_1^A, \ldots, x_t^A)$ and $(x_1^B, \ldots, x_{T-t}^B)$ by decommitting to $\{\mathsf{rv}_i^{(1,A)}\}_{i\in[n]}$ and $\{\mathsf{rv}_i^{(1,B)}\}_{i\in[n]}$.

6. **(Committer's Consistency Proof.)** This stage should be interpreted as C proving consistency between its actions in Steps 1 and 4 (i.e., these two steps commit to the same value) using a WI argument, where the trapdoor statement is that: C manages to commit to a puzzle solu-

tion in Step 4 *either* for **Hard-Puzzle-A** *or* **Hard-Puzzle-B**. Note that this is corresponding to the modification described in Sec. 8.
This step is again conducted in MitH. Note that for the honest committer, the 'effective witness' is the same message m reconstructed from both $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$ and $\{\mathsf{cv}_i^{(2)}\}_{i\in[n]}$, and so the virtual MPC execution in reality evaluates the 'first clause' of F^C_{consis} as defined below.

(a) C prepares n views $\{\mathsf{cv}_i^{(3)}\}_{i\in[n]}$, corresponding to an $(n, 2k)$-MitH execution of the n-party functionality F^C_{consis} described below, where party P_i ($i \in [n]$) uses $\mathsf{cv}_i^{(1)}\|\mathsf{cv}_i^{(2)}$ as input. C commits to each $\mathsf{cv}_i^{(3)}$ ($i \in [n]$) independently in parallel, using Naor's commitment.

- **Functionality F^C_{consis}:** It collects input (and parses it as) $\mathsf{cv}_i^{(1)}\|\mathsf{cv}_i^{(2)}$ from party i for each $i \in [n]$. It then runs the recovery algorithm of VSS to obtain $a \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cv}_1^{(1)}, \ldots, \mathsf{cv}_n^{(1)})$ and $b \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cv}_1^{(2)}, \ldots, \mathsf{cv}_n^{(2)})$. It outputs 1 to each party if
  - $b = a$, **or**
  - b can be parsed as $j\|x'$ such that $j \in [t]$ and $x' = x_j^A$ (recall that x_j^A is among the **Hard-Puzzle-A** solutions revealed by Rrevealed in Step 5, **or**
  - b can be parsed as $j\|x'$ such that $j \in [T-t]$ and $x' = x_j^B$ (recall that x_j^B is among the **Hard-Puzzle-B** solutions revealed by Rrevealed in Step 5.

  Otherwise, it outputs 0 to each party.

(b) **(Trapdoor Coin-Flipping)** C and R then execute the **Coin-Flipping Stage** of the trapdoor coin-flipping protocol shown in Prot. 2, with the trapdoor predicate $\phi(\cdot)$ defined as follows

- **Predicate $\phi(\cdot)$:** It has the values $(x_1^A, \ldots, x_t^A)$ and $(x_1^B, \ldots, x_{T-t}^B)$ hard-wired (recall that these values are revealed in Step 5). On input $j\|x'$, ϕ outputs 1 if and only if *either*
  - $j \in [t]$ and $x' = x_j^A$ *or*
  - $j \in [T-t]$ and $x' = x_j^B$.

  By the completeness of the trapdoor coin-flipping protocol (i.e., Property 2 in Def. 24), at the end of this step, C and R agree on a string η. By a proper choice of length, the string η can be interpreted as specifying a size-k random subset of $[n]$. In the following, we abuse notation by using η to denote the corresponding size-k random subset.

(c) C sends $\{(\mathsf{cv}_i^{(1)}, \mathsf{cv}_i^{(2)}, \mathsf{cv}_i^{(3)})\}_{i\in\eta}$ together with the decommitment information (w.r.t. their respective commitments in Steps 1, 4 and 6a).

(d) R checks the validity of the decommitment information and the consistency among the revealed views $\{(\mathsf{cv}_i^{(1)}, \mathsf{cv}_i^{(2)}, \mathsf{cv}_i^{(3)})\}_{i\in\eta}$. It also checks for each $i \in \eta$ that the output of P_i contained in $\mathsf{cv}_i^{(3)}$ is 1. It aborts if any of these checks fail.

Decommit Stage:

1. C sends $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$ together with the decommitment information w.r.t. the commitments in Step 1.

2. R checks the validity of the decommitment information and the consistency among $\{\mathsf{cv}_i^{(1)}\}_{i\in[n]}$. If these checks are successful, R recovers m as $m \coloneqq \mathsf{VSS}_{\mathsf{Recon}}(\mathsf{cv}_1^{(1)}, \ldots, \mathsf{cv}_n^{(1)})$; otherwise, R rejects and output $\perp$.