A New Approach to Post-Quantum Non-Malleability

Xiao Liang[1], Omkant Pandey[2*], and Takashi Yamakawa[3]

[1] NTT Research, Sunnyvale, USA
xiao.crypto@gmail.com
[2] Stony Brook University, Stony Brook, USA
omkant@cs.stonybrook.edu
[3] NTT Social Informatics Laboratories, Tokyo, Japan
takashi.yamakawa.ga@hco.ntt.co.jp

Abstract. We provide the first *constant-round* construction of post-quantum non-malleable commitments under the minimal assumption that *post-quantum one-way functions* exist. We achieve the standard notion of non-malleability with respect to commitments. Prior constructions required $\Omega(\log^* \lambda)$ rounds under the same assumption.

We achieve our results through a new technique for constant-round non-malleable commitments which is easier to use in the post-quantum setting. The technique also yields an almost elementary proof of security for constant-round non-malleable commitments in the classical setting, which may be of independent interest.

When combined with existing work, our results yield the first constant-round quantum-secure multiparty computation for both classical and quantum functionalities *in the plain model*, under the *polynomial* hardness of quantum fully-homomorphic encryption and quantum learning with errors.

Keywords: Non-Malleable · Post-Quantum · Constant-Round

[*] Supported in part by DARPA SIEVE Award HR00112020026, NSF CAREER Award 2144303, NSF grants 1907908, 2028920, 2106263, and 2128187. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, DARPA, or NSF.

Table of Contents

Table of Contents . . . iii

1 Introduction . . . 1
  1.1 Challenge: Robust Simulatable Extraction . . . 2
  1.2 Our Results . . . 4

2 Technical Overview . . . 5
  2.1 Small-Tag, One-Sided, Synchronous, Classical Setting: Construction . . . 6
  2.2 Proving Non-Malleability . . . 7
  2.3 Small-Tag, One-Sided, Synchronous, Post-Quantum Setting . . . 14
  2.4 Full-Fledged Post-Quantum Non-Malleable Commitments . . . 16
  2.5 Organization . . . 18

3 Preliminaries . . . 18
  3.1 Basic Notations . . . 18
  3.2 Commitment Schemes . . . 19
  3.3 Non-Malleable Commitments . . . 20
  3.4 Extractable Commitments . . . 21
  3.5 Witness-Indistinguishable Proofs of Knowledge . . . 22

4 Small-Tag, One-Sided, Synchronous, Classical Setting . . . 24
  4.1 Construction of $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{OneSided}}$ . . . 25
  4.2 Proving Non-Malleability (Proof of Thm. 2) . . . 26
  4.3 Proof of Lem. 6 . . . 27
  4.4 Proof of Lem. 7 . . . 29
  4.5 Extractor $\mathcal{K}$ (Proof of Lem. 9) . . . 35
  4.6 Proof of Lem. 10 . . . 38
  4.7 Proof of Claim 10 . . . 42
  4.8 Replacing the Injective OWF with Any OWF . . . 43
  4.9 Summarizing the Proof Strategy for Thm. 2 . . . 44

5 Small-Tag, Synchronous, Classical Setting . . . 46
  5.1 High-Level Idea . . . 46
  5.2 Construction of $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{sync}}$ . . . 47
  5.3 Proving Non-Malleability (Proof of Thm. 11) . . . 49
    5.3.1 The Case of $t < \widetilde{t}$ . . . 49
    5.3.2 The Case of $t > \widetilde{t}$ . . . 50

6 Small-Tag, Asynchronous, Classical Setting . . . 50
  6.1 High-Level Idea . . . 50
  6.2 Construction of $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$ . . . 51
  6.3 Adaptive Schedule vs. Predetermined Schedule . . . 54
  6.4 Proving Non-Malleability (Proof of Thm. 13) . . . 55
  6.5 Non-Malleability for Good Schedules . . . 55
    6.5.1 The Case of $t < \widetilde{t}$. . . . 55
    6.5.2 The Case of $t > \widetilde{t}$ . . . 57
  6.6 Non-Malleability for Bad Schedules . . . 58
    6.6.1 Analysis of Bad 1 . . . 59
    6.6.2 Analysis of Bad 2 . . . 59
    6.6.3 Analysis of Bad 3 . . . 61

6.6.4 Analysis of Bad 4 . . . 62
6.6.5 Analysis of Bad 5 . . . 63
7 Extract-and-Simulate Lemma . . . 63
8 Post-Quantum Non-Malleable Commitments . . . 64
8.1 Small-Tag, One-sided, Synchronous, Post-Quantum Setting . . . 64
8.2 Small-Tag, Asynchronous, Post-Quantum Setting . . . 64
8.3 Tag Amplification . . . 67
9 Application: Quantum-Secure MPC in Constant Rounds from QLWE . . . 68
References . . . 71
A Proof of Extract-and-Simulate Lemma (Lem. 20) . . . 72
A.1 Preparation . . . 72
A.2 Proving Lem. 20 . . . 73
B Small-Tag, One-sided, Synchronous, Post-Quantum Setting . . . 77
B.1 Construction . . . 77
B.2 Proving Non-Malleability . . . 78
B.3 Proof of Lem. 27 . . . 79
B.4 Proof of Lem. 28 . . . 81
B.5 Extractor $\mathcal{K}$ (Proof of Lem. 30) . . . 84
B.6 Proof of Lem. 31 . . . 87
B.7 Proof of Claim 34 . . . 91

1 Introduction

Commitments are one of the central primitives in modern cryptography. They are two-party protocols that enable a sender (or committer) to commit to a message to a receiver. It is required that an efficient receiver should learn nothing about the committed message until later the committer chooses to open (or decommit to) it. However, this vanilla promise of "information hiding" does not rule out the following mauling or man-in-the-middle (MIM) attack: An adversary $\mathcal{M}$ could play the role of a receiver in one instance of the commitment (referred to as the *left session*), while simultaneously playing as a committer in another (referred to as the *right session*). In this situation, $\mathcal{M}$ can potentially make the value committed in the right session depend on the value 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 $\mathcal{M}$ could conduct the above attack without explicitly learning the value committed in the left session.

To protect against such attacks, Dolev, Dwork, and Naor [DDN91] introduced the concept of *non-malleable commitments*. Such commitments capture the MIM attack by requiring that in the MIM execution, the *joint distribution* of $\mathcal{M}$'s final output *and* the value committed in the right session is computationally indistinguishable for any values committed by the honest committer in the left session. Of course, $\mathcal{M}$ can always relay without any modifications the messages between the left honest committer and the right honest receiver. This is handled by augmenting the commitment scheme with a *tag*, and $\mathcal{M}$ is considered winning the MIM game only if she does not use the same tag on both sides.[4]

Non-malleable commitments have found several applications in cryptography. They turn out to be a critical ingredient in resolving the exact round complexity of secure multiparty computation in the standard (plain) model [KOS03, PPV08, Wee10, Goy11, GMPP16, BGJ+18, CCG+19], as well as protecting such protocols against concurrent attacks [Can00, Can01, CLOS02, Pas03, PS04, MPR06, BDH+17, CLP20].

Since their introduction, the central question in this area has been the construction of *constant-round* non-malleable commitments under the minimal assumption that one-way functions (OWFs) exist. The original work [DDN91] presented a logarithmic-round construction assuming only OWFs. The works of Barak [Bar02], and Pass and Rosen [PR05] succeeded in obtaining constant-round constructions under the (stronger but standard) assumption of (polynomially-hard) collision-resistant hash functions; this assumption is inherited due to the use of non-black-box simulation techniques [Bar01]. After a long line of follow-up works [PPV08, LPV08, LP09, Wee10], constant-round non-malleable commitments assuming only OWFs were first constructed in independent and concurrent works of Goyal [Goy11] and Lin and Pass [LP11]. Since then, several constructions optimizing various aspects of this primitive, such as the *exact* round complexity [Pas13, GRRV14, GPR16, COSV17, LPS17, Khu17, KS17, BL18, GR19, Khu21] and black-box usage of primitives [Wee10, Goy11, GLOV12], have been proposed, thus achieving an almost complete understanding of this primitive in the *classical* setting where all parties, including the adversary, as well as the communication are classical.

Unfortunately, the results in the classical setting do not usually translate to the quantum setting where one or more parties may be quantum machines. Existing classical techniques often require the ability to rewind and copy the adversaries' code, both of which are not possible in the quantum setting due to the no-cloning theorem [WZ82] and quantum state disturbance [FP96].

These issues, together with the fact that quantum computers might be possible one day, have resulted in a significant push toward developing tools and techniques to reason about security in the

[4] Copying the tag can be shown equivalent to copying the entire interaction assuming one-way functions.

presence of quantum adversaries. Some examples include zero-knowledge [Wat06, BS20], signatures [BZ13], pseudo-random functions [Zha12], secure computation [BCKM21b, ABG$^+$21], and so on.

Post-Quantum Non-Malleable Commitments. A particularly appealing goal in this direction is the construction of the so-called *post-quantum* secure protocols, where the honest parties and their communication channels are entirely *classical* but the adversary is allowed to be a *quantum* (polynomial-time) algorithm. Such protocols have the feature that even if the adversary somehow gains early access to quantum computing capabilities, the honest parties are not required to catch up to remain protected against it.

Agarwal, Bartusek, Goyal, Khurana, and Malavolta [ABG$^+$21]—in their pursuit of constant-round post-quantum secure multiparty computation—construct a constant-round post-quantum non-malleable commitments w.r.t. the special case of *synchronous schedules* where, upon receiving a message in the left session, $\mathcal{M}$ must respond with the corresponding message of the right session immediately, in reach round of the protocol. They achieve this assuming super-polynomial quantum hardness of the learning with errors (QLWE) problem.

In the general (i.e., asynchronous) setting, the first positive result was recently obtained by Bitansky, Lin, and Shmueli [BLS22] who, in the post-quantum setting, construct a $\log^*(\lambda)$-round protocol assuming only post-quantum OWFs. They present a general compiler to convert any k-round (ε-simulatable) post-quantum extractable commitment to $k^{O(1)} \cdot \log^*(\lambda)$-round post-quantum non-malleable commitments, and then rely on the very recent work of Chia, Chung, Liang, and Yamakawa [CCLY22] where such an extractable commitment protocol is constructed in constant rounds from only post-quantum OWFs (see also [CCLY22, Section 1.2]). This brings us tantalizingly close to constant rounds. However, the techniques in [BLS22] rely on scheduling and amplification techniques, which inherently require non-constant rounds to support (the standard requirement of) large tags or identities. It is unclear if these techniques can yield constant rounds.

The central question in this area thus still remains open:

> **Question 1:** *Do constant-round post-quantum non-malleable commitments, assuming only post-quantum one-way functions, exist?*

1.1 Challenge: Robust Simulatable Extraction

Toward answering **Question 1**, let us first discuss about the challenges.

As mentioned above, proving non-malleability of a commitment scheme requires one to show that the *joint distribution* of the final state of the MIM $\mathcal{M}$ (denoted by $\mathsf{st}_\mathcal{M}$) and the value committed in the right session (denoted by $\widetilde{m}$) are computationally indistinguishable when the left-session committed value changes from any m_0 to any $m_1 \neq m_0$. Typically, this is done by a careful design of a sequence of hybrids, where the left value is gradually changed from m_0 to m_1; And non-malleability will then be established by showing that the joint distribution of $(\mathsf{st}_\mathcal{M}, \widetilde{m})$ is indistinguishable between each pair of adjacent hybrids. As in many other cryptographic proofs, one usually needs to reduce the indistinguishability between adjacent hybrids to some computational hardness assumptions. But this step is especially hard for non-malleability proofs due to the following "inefficient testability" issue—The value $\widetilde{m}$ is hidden in the transcript of the right interaction, and no efficient machine could obtain it. Thus, the event that "the joint distribution of $(\mathsf{st}_\mathcal{M}, \widetilde{m})$ changes" is not efficiently testable, thus forbidding an efficient reduction to the underlying hardness assumptions.

The most common template to address the above issue is to (efficiently) extract the value $\widetilde{m}$ from some "extractable gadget" (e.g., extractable commitments or proofs of knowledge) in the right session. The hope is: if the extracted value, denoted by $\widetilde{m}'$, is equal to $\widetilde{m}$ with good-enough probability, then one can conduct the above reduction using $(\mathsf{st}_{\mathcal{M}}, \widetilde{m}')$ in place of $(\mathsf{st}_{\mathcal{M}}, \widetilde{m})$. That is, since the extraction of $\widetilde{m}'$ is efficient, it saves the reduction from the aforementioned inefficient testability issue.

To properly implement this template, it is crucial to maintain the following conditions (the combination of which we call *Robust Simulatable Extractability.*)

– **Extractability:** One can extract the committed message $\widetilde{m}$ in the right from some extractable gadget;

– **Simulatability:** One can simulate $\mathcal{M}$'s final (i.e., post-extraction) state while extracting $\widetilde{m}$;

– **Robustness:** The extraction of $\widetilde{m}$ (in the right session) does not harm the hiding property of the left session (or some left-session gadget on whose hiding property the security reduction relies).

Roughly speaking, previous designs of non-malleable protocols (in the classical setting) can be thought of as developing different (and better-and-better) techniques that enable the above robust simulatable extractability.

However, robust simulatable extractability turns out to be hard to obtain *in the post-quantum setting*. First, as mentioned earlier, special techniques are needed to perform simulation for quantum adversaries due to the no-cloning theorem; Extracting a desired value while simultaneously simulating the quantum adversaries' internal state is even harder. Fortunately, the recent works [CCLY22, LMS21] did provide a constant-round solution to simulatable extraction based solely on post-quantum OWFs.

However, the picture becomes unclear when we additionally require robustness. Known simulatable extraction techniques treat the adversary as a single reversible operation (i.e., unitary) and "rewind" it coherently. However, if the adversary talks in straight-line with the external left committer C (that cannot be rewound), those techniques seem inapplicable. Actually, this robustness issue already appeared in the classical setting. But we would like to point out that the quantum power of adversaries complicates it further: It is reasonable to expect that the extractor (to be constructed) may need to "read" the messages exchanged between the MIM adversary $\mathcal{M}$ and the left committer C. However, known quantum rewinding strategies need to treat the adversary as a reversible operation. When we view $(C, \mathcal{M})$ as a joint adversary to perform quantum rewinding, it is unclear if the simulator can "read" (technically, *measure*) the messages exchanged between C and $\mathcal{M}$, because this may irreversibly collapse the internal quantum state of the joint $(C, \mathcal{M})$ adversary. Therefore, it is unclear if existing post-quantum rewinding techniques could be used in the MIM setting when robustness is a concern. This indeed represents the major difficulty when one tries to build constant-round post-quantum non-malleable commitments by quantizing the security proofs of the classical ones (e.g., [Goy11, LP11, GLOV12, GRRV14, GPR16, COSV17, GR19]).

Existing Techniques. The two recent works mentioned earlier demonstrated possible solutions to the robust simulatable extractability issue, if one is willing to make stronger hardness assumptions, *or* does not insist on constant rounds:

– [BLS22] took a similar approach as in [DDN91]. Roughly, the idea is to introduce enough rewinding opportunities for extraction (usually referred to as "slots") in the construction such that there is always a "free slot", namely, a slot that does not interleave with any messages exchange between $\mathcal{M}$ and the left committer C (who will become the external challenger of the hardness-providing gadget in the security proof); Then, one can extract $\widetilde{m}$ from this free slot using known post-quantum rewinding strategies. This approach is unlikely to give a constant-round construction (even in the classical setting). The constructions from [BLS22] require at least $\log^*(\lambda)$ rounds.

– [ABG$^+$21] took a similar approach as in [PW10, KS17], which essentially obtained robust simulatable extractability via *complexity leveraging*. At a high level, the idea is to hide the committed value in some computationally hard gadget, whose hardness depends on the tag. This hard gadget is cleverly designed to admit the following strategy: In the MIM execution, because of the asymmetry of the left and right tags (i.e., $t \neq \widetilde{t}$), the reduction can extract $\widetilde{m}$ by breaking the *right* hard gadget via brute force in some slightly super-polynomial time $T(\lambda)$; However, the MIM adversary cannot break the *left* hard gadget in time $\mathsf{poly}\big(T(\lambda)\big)$. Since the extraction in this template is conducted via brute force, the robust simulatable extraction issue can be circumvented. However, this approach is unlikely to work without super-polynomial hardness assumptions due to the use of complexity leveraging.

1.2 Our Results

In this work, we answer **Question 1** affirmatively by providing the first constant-round construction of post-quantum non-malleable commitments assuming only post-quantum one-way functions.

As discussed above, major classical approaches to OWF-based constant-round non-malleable commitments seem not to be "quantum-friendly." Therefore, we first propose a new classical construction whose security proof is quantum-friendly. That is, when designing the new protocol, we restrict ourselves to techniques that avoid state-cloning and the above robust simulatable extractability issue. We find this result already interesting as it improves the diversity of the approaches known in the classical setting.

Next, we show that our new classical construction can be made quantum-secure once its building blocks are replaced by their post-quantum counterparts. This is possible because our classical construction is deliberately designed to be quantum-friendly.

Theorem 1 (Informal). *Assuming the existence of post-quantum one-way functions, there exists a constant-round construction of post-quantum non-malleable commitments.*

Thm. 1 yields interesting corollaries w.r.t. *multi-party computation* (MPC) in the quantum era. For classical functionalities, the recent work [ABG$^+$21] presents the first post-quantum MPC in constant rounds, from a mildly super-polynomial *quantum hardness of Learning with Errors* (QLWE) assumption and a QLWE-based circular security assumption. They need the super-polynomial hardness of QLWE (only) to build constant-round post-quantum non-malleable commitments, which serve as a building block to their MPC. Plugging our non-malleable commitment into their framework yields the following result:

Corollary 1 (Informal). *Assuming (polynomial) QLWE and the QLWE-based circular security assumption (as in [ABG$^+$21]), there exists a constant-round construction of post-quantum MPC for classical functionalities, i.e., an MPC protocol secure against QPT adversaries where honest parties only need to perform classical computation and communication.*

For quantum functionalities, the recent work [BCKM21a] presents a constant-round quantum-secure MPC for quantum functionalities *in the CRS model*, based on the hardness of QLWE. It is easy to see that Corollary 1 provides a constant-round implementation for the CRS required by the [BCKM21a] protocol. This observation leads to the first constant-round quantum-secure MPC for quantum functionalities from polynomial hardness assumptions *without any trusted setup*.

Corollary 2 (Informal). *Assuming (polynomial) QLWE and the QLWE-based circular security assumption (as in [ABG$^+$21]), there exists a constant-round construction of quantum-secure MPC for quantum functionalities.*

2 Technical Overview

We will first present a construction $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ that is non-malleable *in the classical setting* with the following restrictions:

- **Small-Tag:** It only supports tags from the polynomial-size space $[n] \coloneqq \{1, \ldots, n(\lambda)\}$, where $n(\lambda)$ is a polynomial on the security parameter λ;
- **One-Sided:** Its non-malleability holds only if, in the MIM game, the left-session tag t is smaller than the right-session tag $\widetilde{t}$.
- **Synchronous:** Its non-malleability holds only in the synchronous setting. This refers to the setting where upon receiving a message in the right (resp. left) session, the MIM adversary immediately responds with the corresponding message in the left (resp. right) session.

It is a common approach in the literature to first obtain a construction under the above conditions, and then convert it to a full-fledged non-malleable commitment. While there exist standard techniques that take care of the latter step, the former step (i.e., constructing $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$) is typically where difficulty lies.

In the classical setting, once we obtain $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$, we can apply known techniques to remove the **One-Sided** restriction, yielding a small-tag, synchronous protocol, which we denote as $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$. Then, known compilers (e.g., [Wee10]) can be used to remove the **Small-Tag** and **Synchronous** restrictions *at one stroke*. This leads to a full-fledged non-malleable commitment $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{TG}}$ in the classical setting.

We emphasize that all the above protocols are non-malleable *only in the classical setting*. However, they are designed deliberately using quantum-friendly techniques, which makes it possible to quantize their security proofs. We elaborate on that in the sequel.

Post-Quantum Tag Amplification. Recall that our eventual goal is to achieve *post-quantum* non-malleability. The use of quantum-friendly techniques will allow us to prove *post-quantum* non-malleability of $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$. Also, the aforementioned (classical) conversion from $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ to $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$ extends naturally to the post-quantum setting as well. However, the classical compiler from $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$ to $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{TG}}$ seems not to extend to the post-quantum setting.

This has already been observed in [BLS22]. The authors of [BLS22] addressed this issue by constructing a new tag amplifier that converts a small-tag, asynchronous post-quantum non-malleable commitment to a large-tag (i.e, $t \in [2^\lambda]$), asynchronous post-quantum non-malleable commitment. But it is worth noting that this tag amplifier requires the small-tag protocol to be *asynchronously* secure; This is in contrast to the aforementioned classical compiler, which handles asynchronicity and tag-size amplification at one stroke.

To overcome this problem, we will first show that our post-quantum version of $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$ can be modified to achieve non-malleability in the asynchronous setting, yielding a protocol $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg,PQ}}$, which is exactly a small-tag, asynchronously non-malleable commitment in the post-quantum setting. Now, the [BLS22] tag amplifier can be applied to $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg,PQ}}$, leading to the full-fledged post-quantum non-malleable commitment $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{TG,PQ}}$ we want.

Organization of Technical Overview. In Sec. 2.1 and 2.2, we overview the main idea behind our construction of $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ in the classical setting. This is the most technically involved part where the main difficulty lies.

As mentioned earlier, protocol $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ is designed using only quantum-friendly techniques. Thus, the proof of its non-malleability extends to the post-quantum setting once we replace its building blocks by their post-quantum counterparts. (We provide a high-level explanation for that on Page 12.) We then show how this can be done in Sec. 2.3, where we obtain the one-sided, small-tag, synchronous, *post-quantum* non-malleable commitment $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg,PQ}}$.

Finally, we provide in Sec. 2.4 a brief overview on how to convert $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg,PQ}}$ to the full-fledged post-quantum non-malleable commitment $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{TG,PQ}}$, achieving our eventual goal.

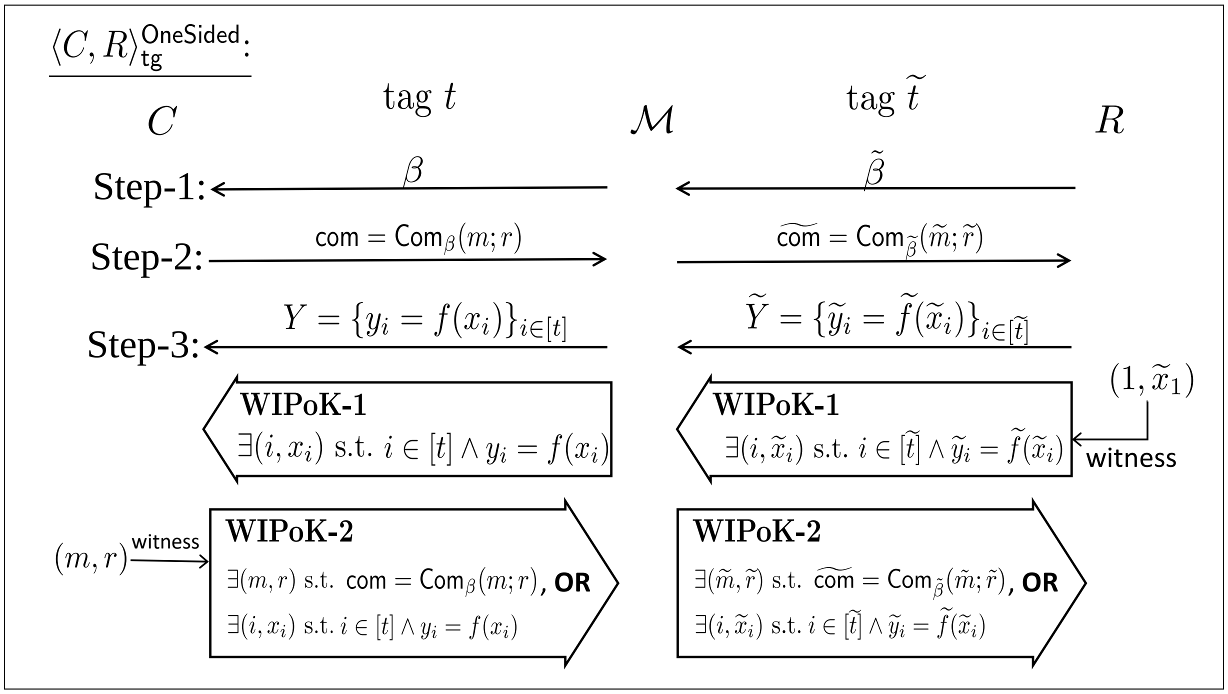
2.1 Small-Tag, One-Sided, Synchronous, Classical Setting: Construction

Our construction of $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ is easy to describe. It supports tags from the space $[n(\lambda)]$, where $n(\lambda)$ is any (fixed) polynomial on the security parameter λ.

To proceed with a tag $t \in [n]$, we first ask the committer C to commit to m using a statistically binding scheme $\mathsf{com} = \mathsf{Com}(m; r)$ (e.g., Naor's commitment). Then, the receiver R sends a *non-interactive hard puzzle* that has exactly t distinct solutions; R also gives a witness-indistinguishable proof of knowledge[5] (referred to as **WIPoK-1**) to prove that he knows one of the t solutions. Finally, C is required to prove using another WIPoK (referred to as **WIPoK-2**) that he knows *either* the value committed in com *or* one solution to R's hard puzzle.

We depict this construction (in the MIM setting) in Fig. 1, where the "hard puzzle" is implemented with the problem of "finding one of the preimages of the t OWF images $\{y_i\}_{i\in[t]}$". Throughout this overview, we assume that the OWF f is injective (otherwise, this "hard puzzle" may have more than t solutions). But this is only to ease the presentation—We will describe in Sec. 4.8 a simple trick to remove this injectivity requirement.

The main intuition underlying this construction is best illustrated by the proof of its non-malleability, which we show next in Sec. 2.2.



$\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$:
C
tag t
$\mathcal{M}$
tag $\tilde{t}$
R
Step-1:
β
$\tilde{\beta}$
Step-2:
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\tilde{\beta}}(\tilde{m}; \tilde{r})$
Step-3:
$Y = \{y_i = f(x_i)\}_{i\in[t]}$
$\tilde{Y} = \{\tilde{y}_i = \tilde{f}(\tilde{x}_i)\}_{i\in[\tilde{t}]}$
WIPoK-1
$\exists(i, x_i)$ s.t. $i \in [t] \wedge y_i = f(x_i)$
$\exists(i, \tilde{x}_i)$ s.t. $i \in [\tilde{t}] \wedge \tilde{y}_i = \tilde{f}(\tilde{x}_i)$
$(1, \tilde{x}_1)$
witness
WIPoK-2
(m, r)
$\exists(m, r)$ s.t. $\mathsf{com} = \mathsf{Com}_\beta(m; r)$, OR
$\exists(i, x_i)$ s.t. $i \in [t] \wedge y_i = f(x_i)$
$\exists(\tilde{m}, \tilde{r})$ s.t. $\widetilde{\mathsf{com}} = \mathsf{Com}_{\tilde{\beta}}(\tilde{m}; \tilde{r})$, OR
$\exists(i, \tilde{x}_i)$ s.t. $i \in [\tilde{t}] \wedge \tilde{y}_i = \tilde{f}(\tilde{x}_i)$

Fig. 1: Man-in-the-Middle Execution of $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$

[5] Indeed, a WI *argument* of knowledge suffices (see Sec. 2.3).

2.2 Proving Non-Malleability

Instead of giving out the security proof directly, we start with a naive attempt that will fail. It will reveal the main difficulty in the proof of non-malleability and how our design addresses it.

The First Attempt. In the MIM execution shown in Fig. 1, imagine that we invoke the knowledge extractor[6] for the right **WIPoK-2** to extract a witness w'. It is easy to see that the only possible values for w' is $(\widetilde{m}, \widetilde{r})$ or $(1, \widetilde{x}_1)$, because they are the only information that $\mathcal{M}$ could potentially possess assuming it cannot break the one-wayness of the OWF $\widetilde{f}$ in the right **Step-3**. Furthermore, we know for sure that w' can only be $(\widetilde{m}, \widetilde{r})$ because of the WI property of the right **WIPoK-1** and the following (standard) argument: Imagine that we switch to $(2, \widetilde{x}_2)$ as the witness to finish the right **WIPoK-1**. By the WI property of the right **WIPoK-1**, we know that the distribution of the extracted w' cannot change (up to negligible probability). That is, *if w' could take the value $(1, \widetilde{x}_1)$ before we switch to $(2, \widetilde{x}_2)$ in the right* **WIPoK-1***, it would keep taking the value $(1, \widetilde{x}_1)$ when we are using $(2, \widetilde{x}_2)$ in the right* **WIPoK-1**. But this cannot happen as it breaks the one-wayness of $\widetilde{f}$ on the image $\widetilde{y}_1$ using a standard reduction—Consider an external challenger for the one-wayness of $\widetilde{f}$; The challenger sends us a challenge y^*; We run the game shown in Fig. 1, while using y^* in place of $\widetilde{y}_1$ and using $(2, \widetilde{x}_2)$ as the witness to perform the right **WIPoK-1**. If we finally extract $w' = (1, x_1)$, we find a preimage for y^*.

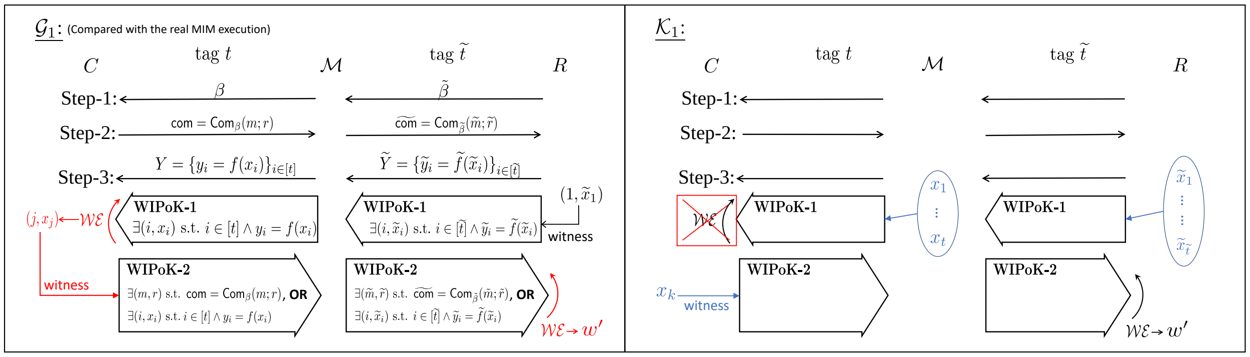
With the above observation, we *hope to* prove non-malleability in the following way: We change the left committed value from m_0 to m_1, while extracting the right committed value $w' = (\widetilde{m}, \widetilde{r})$ using the knowledge extractor for the right **WIPoK-2**. If $\mathcal{M}$ really makes $\widetilde{m}$ depend on the left committed value, then the reduction can detect this change by checking the extracted w'. This breaks the hiding property[7] of the left execution when the left committed value changes from m_0 to m_1. However, this naive idea will not work: When we invoke the knowledge extractor for the right **WIPoK-2**, $\mathcal{M}$ also rewinds the left **WIPoK-2**. So, the left interaction is not "in straight-line" anymore. Thus, we could not build the reduction to the hiding property of the left interaction.

Reduction to Naor's Commitment? Though the above attempt does not work, it inspires the following thoughts: What if we reduce non-malleability to the hiding property of Naor's commitment in the left **Step-1** and **Step-2**? There is at least some hope as the left Naor's commitment is not interleaved with the right **WIPoK-2** (recall that we are in the synchronous setting). That is, we can generate the **Step-1** and **Step-2** messages in the left interaction by forwarding messages between $\mathcal{M}$ and an (external) challenger for the hiding property of Naor's commitment; Meanwhile, we extract $w' = (\widetilde{m}, \widetilde{r})$ from the right **WIPoK-2** as before. Then, if $\mathcal{M}$ makes the value $\widetilde{m}$ depend on the value committed in the first two steps in the left, we win the hiding game by checking the extracted w'.

To implement this idea, we first need to make the interaction happening after the left **Step-2** *independent of* m; Otherwise, the above reduction will not work—Because the left Naor's commitment is now coming from an external challenger so that the reduction does not posses the value m anymore, which is required to finish the remaining steps (in particular, the left **WIPoK-2**) of the left interaction. To address this issue, we consider an intermediate execution $\mathcal{G}_1$ shown in Fig. 2a—The only difference (shown in red) between $\mathcal{G}_1$ and the real MIM game (Fig. 1) is that $\mathcal{G}_1$ uses the knowledge extractor $\mathcal{WE}$ in the left **WIPoK-1**, and uses the extracted witness (j, x_j)

[6] We remark that there exist different definitions for *Proofs of Knowledge*. Throughout this paper, we use Lindell's formalism called *Witness Extended Emulation* [Lin03]. This definition ensures the existence of a knowledge extractor $\mathcal{WE}$ such that for any P^* that convinces V with probability p, $\mathcal{WE}^{P^*}$ will extract a valid witness with probability $\geq p - \mathsf{negl}(\lambda)$. See Sec. 3.5 for more details.

[7] It is easy to see that our construction, in the stand-alone setting, is a computationally hiding commitment.



$\mathcal{G}_1$: (Compared with the real MIM execution)
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
Step-1:
β
$\widetilde{\beta}$
Step-2:
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
Step-3:
$Y = \{y_i = f(x_i)\}_{i \in [t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i \in [\widetilde{t}]}$
WIPoK-1
$\exists (i, x_i)$ s.t. $i \in [t] \wedge y_i = f(x_i)$
$\exists (i, \widetilde{x}_i)$ s.t. $i \in [\widetilde{t}] \wedge \widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)$
$(j, x_j) \leftarrow \mathcal{WE}$
$(1, \widetilde{x}_1)$
witness
WIPoK-2
$\exists (m, r)$ s.t. $\mathsf{com} = \mathsf{Com}_\beta(m; r)$, OR
$\exists (i, x_i)$ s.t. $i \in [t] \wedge y_i = f(x_i)$
$\exists (\widetilde{m}, \widetilde{r})$ s.t. $\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$, OR
$\exists (i, \widetilde{x}_i)$ s.t. $i \in [\widetilde{t}] \wedge \widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)$
$\mathcal{WE} \rightarrow w'$
$\mathcal{K}_1$:
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
Step-1:
Step-2:
Step-3:
$\mathcal{WE}$
WIPoK-1
x_1 ⋮ x_t
$\widetilde{x}_1$ ⋮ $\widetilde{x}_{\widetilde{t}}$
WIPoK-2
x_k
witness
$\mathcal{WE} \rightarrow w'$
(a)
(b)

Fig. 2: Machines $\mathcal{G}_1$ and $\mathcal{K}_1$

as the witness to go through the left **WIPoK-2**. In this way, the interaction after the left **Step-2** does not depend on m anymore. However, this introduces new problems: We can no longer make use of the WI property of the right **WIPoK-1** as before, because this $\mathcal{WE}$ needs to rewind the left **WIPoK-1**, resulting in rewindings of the right **WIPoK-1** as well (since we are in the **Synchronous** setting). Thus, we can no longer argue that the extracted w' must be $(\widetilde{m}, \widetilde{r})$ as in **The First Attempt** part.

Actually, there is a deeper reason why this naive attempt is bound to fail in establishing $w' = (\widetilde{m}, \widetilde{r})$. When we run $\mathcal{WE}$ for the left **WIPoK-1**, $\mathcal{M}$ could also learn the witness $(1, \widetilde{x}_1)$ used by R in the right **WIPoK-1**. When we switch the witness from (m, r) to the extracted (j, x_j) in the left **WIPoK-2**, nothing stops $\mathcal{M}$ from switching her witness from $(\widetilde{m}, \widetilde{r})$ to $(1, \widetilde{x}_1)$. Therefore, we can no longer argue that the extracted w' must be $(\widetilde{m}, \widetilde{r})$. We emphasize that the WI property of the left **WIPoK-2** does not help in ruling out this possibility—*WI does not protect against a man-in-the-middle adversary who is trying to make the right WIPoK instance depend on the left WIPoK instance.* Instead, this is a "non-malleability" type of requirement rather than (plain) witness indistinguishability. In that sense, the above argument did not really address the issue of non-malleability; Rather, it simply "pushed" the non-malleability requirement to the WIPoK used in the final **WIPoK-2** stage.

Thus, new ideas seem necessary to enforce non-malleability. This leads us to the real virtue of our design—In the sequel, we will demonstrate that the t-solution hard puzzle introduces "asymmetry" between the right and the left sessions as $t < \widetilde{t}$; This will allow us to perform a pigeon-hole-style argument, which can be used to address the heart of the non-malleability problem.

A Pigeon-Hole Argument. To address the above issue, we consider another hybrid $\mathcal{K}_1$ shown in Fig. 2b. Hybrid $\mathcal{K}_1$ is similar to $\mathcal{G}_1$ but it does not rewind the left **WIPoK-1** for witness extraction (we will explain shortly which witness $\mathcal{K}_1$ will use to perform the left **WIPoK-2**).

We first make a crucial observation: There are $\widetilde{t}$ preimages $\{\widetilde{x}_1, \ldots, \widetilde{x}_{\widetilde{t}}\}$ that R could potentially use as the witness to perform the right **WIPoK-1**. In contrast, there are at most t preimages $\{x_1, \ldots, x_t\}$ that $\mathcal{M}$ could potentially use as the witness to perform the left **WIPoK-1**. Since the **WIPoK-1** stage itself is not "non-malleable", it is possible that $\mathcal{M}$'s witness used there depends on R's witness used in the right **WIPoK-1**.[8] For example, if R uses $\widetilde{x}_1$ to perform the right **WIPoK-1**, $\mathcal{M}$ may use, say, x_3 in the left **WIPoK-1**; If R instead uses $\widetilde{x}_2$, $\mathcal{M}$ may switch to x_5 in the left **WIPoK-1**.

[8] Again, standard witness-indistinguishability does not rule out such dependency.

Now, recall that $t \leq \widetilde{t} - 1$ (because we have the **One-Sided** condition). It then follows from the pigeon-hole principle that there must exist a distinct pair $\widetilde{x}_i, \widetilde{x}_j \in \{\widetilde{x}_1, \ldots, \widetilde{x}_{\widetilde{t}}\}$ and an $x_k \in \{x_1, \ldots, x_t\}$ such that $(\widetilde{x}_i, \widetilde{x}_j, x_k)$ form the following correspondence: *No matter R uses $\widetilde{x}_i$ or $\widetilde{x}_j$ in the right* **WIPoK-1***, $\mathcal{M}$ always uses (the same) x_k in the left* **WIPoK-1**.

If we assume that $\mathcal{K}_1$ somehow "magically" knows this pigeon-hole tuple $(\widetilde{x}_i, \widetilde{x}_j, x_k)$, then we can prove $w' = \widetilde{m}$ using the following argument. Consider two scenarios:

1. When R uses $\widetilde{x}_i$ in the right **WIPoK-1** and C (or $\mathcal{K}_1$) uses x_k in the left **WIPoK-2**, we can prove that the extracted w' can only take the values of $\widetilde{m}$ or $\widetilde{x}_i$. Intuitively, this is because $\widetilde{m}$ and $\widetilde{x}_i$ are the only witnesses that $\mathcal{M}$ could potentially use for the right **WIPoK-2**, assuming it cannot break the one-wayness of the right OWF $\widetilde{f}$. We suppress the full proof in this informal discussion, as we will show a formal (and slightly different) argument in the next subsection.

2. When R uses $\widetilde{x}_j$ in the right **WIPoK-1** and C uses (the same) x_k in the left **WIPoK-2**, we can prove that the extracted w' can only take the values of $\widetilde{m}$ or $\widetilde{x}_j$. This follows from a similar argument as in the above bullet.

Next, observe that $\mathcal{K}_1$ does *not* rewind the **WIPoK-1** stage. By the witness indistinguishability of the right **WIPoK-1**, it follows that the witness used by $\mathcal{M}$ in the right **WIPoK-2** cannot change when R switches between $\widetilde{x}_i$ and $\widetilde{x}_j$ in the right **WIPoK-1** (otherwise, we can invoke the knowledge extractor of the right **WIPoK-2** to extract the used witness to detect this change). This, together with the above two bullets, implies that the extracted w' can only take the value $\widetilde{m}$ if R uses $\widetilde{x}_i$ (or $\widetilde{x}_j$) in the right **WIPoK-1**. Notice that in this argument, we do not rely on the "non-malleability" (or even WI) of the final **WIPoK-2** stage, because in both scenarios described above, C uses the same x_k in the left **WIPoK-2**.

Although the above reasoning looks promising, it suffers from the following obstacles:

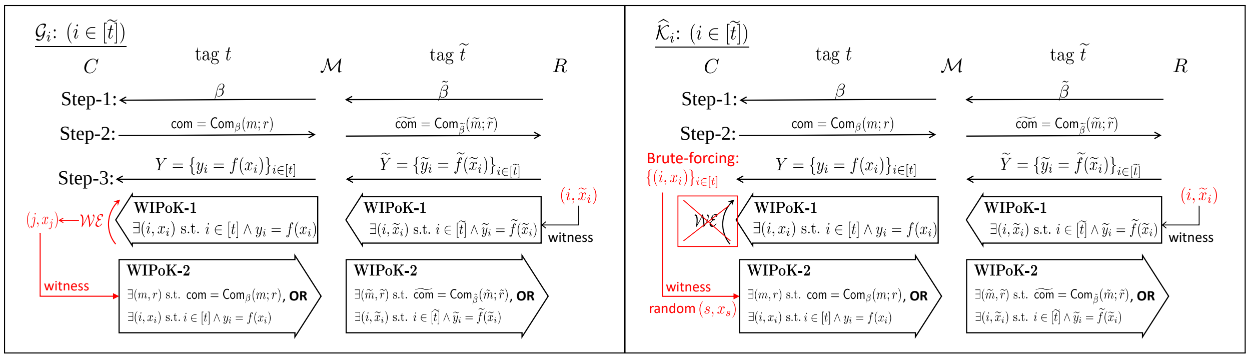
1. In the above, we assumed that $\mathcal{K}_1$ "magically" knows the pigeon-hole tuple $(\widetilde{x}_i, \widetilde{x}_j, x_k)$. However, it is unclear how $\mathcal{K}_1$ could (efficiently) learn this tuple. Note that $\mathcal{K}_1$ cannot try to learn x_k by rewinding (i.e., running the knowledge extractor for) the left **WIPoK-1**; Otherwise, we will be back to hybrid $\mathcal{G}_1$ and cannot rely on the WI property of the right **WIPoK-1** (as we did in the previous paragraph).

2. The above pigeon-hole correspondence is not accurate. In fact, it is possible that $\mathcal{M}$ changes the *distributions* (instead of *concrete values*, as in the previous discussion) of the witness used in the left **WIPoK-1**. For example, when R uses $\widetilde{x}_1$ in the right **WIPoK-1**, $\mathcal{M}$ samples a witness for the left **WIPoK-1** uniformly at random from $\{x_1, \ldots, x_t\}$; While R switches to $\widetilde{x}_2$, $\mathcal{M}$ may decide to sample a witness according to the Gaussian distribution instead. In this case, even with unbounded computational power, it is unclear how to identify (or define) the tuple $(\widetilde{x}_i, \widetilde{x}_j, x_k)$.

Nevertheless, this hybrid $\mathcal{K}_1$ divulges the usefulness of the t-solution hard puzzle. It provides an approach to argue $w' = \widetilde{m}$, thus having a potential to help us establish non-malleability. In the following, we show our main technical lemma which overcomes the above two obstacles, allowing us to formalize the above argument properly.

The Main Technical Lemma. Let us summarize what we have so far. We described two hybrids: Hybrid $\mathcal{G}_1$ seems to help us reduce non-malleability to the computational-hiding property of the left Naor's commitment. But it actually pushes the non-malleability requirement to the final **WIPoK-2** stage (or more technically, we are not able to prove that the extracted w' is equal to the right-side committed value $\widetilde{m}$). In contrast, hybrid $\mathcal{K}_1$ does not suffer from this issue. It instead relies on a pigeon-hole-style argument, which employs the structure of the t-solution hard puzzle to enforce non-malleability. However, to formalize this argument, one has to (i) figure out how $\mathcal{K}_1$ can learn the pigeon-hole tuple (Obstacle 1), and (ii) perform the pigeon-hole argument in a "distributional" manner (Obstacle 2).

In the following, we present our main technical lemma. This lemma can be understood as a combination of $\mathcal{G}_1$ and $\mathcal{K}_1$—In its proof, we will define analogs of these two hybrids (i.e., the $\{\mathcal{G}_i\}_{i\in[\widetilde{t}]}$ and $\{\widehat{\mathcal{K}}_i\}_{i\in[\widetilde{t}]}$ in Fig. 3) and "jump between" them to take advantage of both hybrids, while performing the pigeon-hole argument in a "distributional" fashion.


$\mathcal{G}_i$: $(i \in [\widetilde{t}])$
tag t
tag $\widetilde{t}$
C
$\mathcal{M}$
R
Step-1:
β
$\widetilde{\beta}$
Step-2:
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
Step-3:
$Y = \{y_i = f(x_i)\}_{i\in[t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i\in[\widetilde{t}]}$
$(j, x_j) \leftarrow \mathcal{WE}$
WIPoK-1
$\exists(i, x_i)$ s.t. $i \in [t] \wedge y_i = f(x_i)$
WIPoK-1
$\exists(i, \widetilde{x}_i)$ s.t. $i \in [\widetilde{t}] \wedge \widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)$
$(i, \widetilde{x}_i)$
witness
witness
WIPoK-2
$\exists(m, r)$ s.t. $\mathsf{com} = \mathsf{Com}_\beta(m; r)$, OR
$\exists(i, x_i)$ s.t. $i \in [t] \wedge y_i = f(x_i)$
WIPoK-2
$\exists(\widetilde{m}, \widetilde{r})$ s.t. $\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$, OR
$\exists(i, \widetilde{x}_i)$ s.t. $i \in [\widetilde{t}] \wedge \widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)$
$\widehat{\mathcal{K}}_i$: $(i \in [\widetilde{t}])$
tag t
tag $\widetilde{t}$
C
$\mathcal{M}$
R
Step-1:
β
$\widetilde{\beta}$
Step-2:
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
Brute-forcing:
$\{(i, x_i)\}_{i\in[t]}$
$Y = \{y_i = f(x_i)\}_{i\in[t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i\in[\widetilde{t}]}$
$\cancel{\mathcal{WE}}$
WIPoK-1
$\exists(i, x_i)$ s.t. $i \in [t] \wedge y_i = f(x_i)$
WIPoK-1
$\exists(i, \widetilde{x}_i)$ s.t. $i \in [\widetilde{t}] \wedge \widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)$
$(i, \widetilde{x}_i)$
witness
witness
random (s, x_s)
WIPoK-2
$\exists(m, r)$ s.t. $\mathsf{com} = \mathsf{Com}_\beta(m; r)$, OR
$\exists(i, x_i)$ s.t. $i \in [t] \wedge y_i = f(x_i)$
WIPoK-2
$\exists(\widetilde{m}, \widetilde{r})$ s.t. $\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$, OR
$\exists(i, \widetilde{x}_i)$ s.t. $i \in [\widetilde{t}] \wedge \widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)$

(a) (b)

Fig. 3: Machines $\mathcal{G}_i$ and $\widehat{\mathcal{K}}_i$ (Difference is highlighted in red color)

First, we define machines $\mathcal{G}_i$ for each $i \in [\widetilde{t}]$ (as shown in Fig. 3a). Machine $\mathcal{G}_i$ for $i \neq 1$ behaves identically to the $\mathcal{G}_1$ described earlier (Fig. 2a), except that $\mathcal{G}_i$ uses $(i, \widetilde{x}_i)$ as the witness when performing the right **WIPoK-1**. Again, we define the *extracted witness* w' to be the witness extracted by running the knowledge extractor for the right **WIPoK-2** in $\mathcal{G}_i$. Then, we show the main technical lemma, which we will use to establish non-malleability later.

Lemma 1 (Informal Version of Lem. 9). *Assume that* in the real MIM execution*, $\mathcal{M}$ convinces the honest right receiver R with some noticeable probability $p(\lambda)$. Then, there exists some $i \in [\widetilde{t}]$ such that the extracted witness w' in $\mathcal{G}_i$ must be a valid opening $(\widetilde{m}, \widetilde{r})$ for $\widetilde{\mathsf{com}}$ with another noticeable probability $p'(\lambda)$.*

Proof of Lem. 1. We prove Lem. 1 by contradiction. In the following, we assume for contradiction that for all $i \in [\widetilde{t}]$, the extracted w' contains a valid opening to $\widetilde{\mathsf{com}}$ with at most *non-noticeable* probability. (For simplicity, we ignore the difference between "non-noticeable" and "negligible" and use the term "negligible" in place of "non-noticeable" in this proof.)

First, we claim that in game $\mathcal{G}_i$ (for any $i \in [\widetilde{t}]$), R must also be convinced with probability $p \pm \mathsf{negl}(\lambda)$. This claim follows from the PoK property of the left **WIPoK-1** and the WI property of the right **WIPoK-1** and the left **WIPoK-2**. Since this proof is rather standard, we do not elaborate on it in this overview. More details can be found in Sec. 4.5.

Next, we observe that in $\mathcal{G}_i$, the extracted witness w' can only take the values $(\widetilde{m}, \widetilde{r})$ or $(i, \widetilde{x}_i)$; Otherwise, we break the one-wayness of the right OWF $\widetilde{f}$ (this follows from the same argument we made earlier). However, our assumption (for contradiction) says that in each $\mathcal{G}_i$, $w' \neq (\widetilde{m}, \widetilde{r})$ except for with negligible probability. Therefore, we obtain the following inequality:

$$\forall i \in [\widetilde{t}], \ \left|\Pr\left[w' = (i, \widetilde{x}_i) \text{ in } \mathcal{G}_i\right] - p\right| \leq \mathsf{negl}(\lambda). \quad (1)$$

Next, we define $\{\widehat{\mathcal{K}}_i\}_{i\in[\widetilde{t}]}$ (shown in Fig. 3b)[9]. For each $i \in [\widetilde{t}]$, $\widehat{\mathcal{K}}_i$ behaves identically as $\mathcal{G}_i$ except that $\widehat{\mathcal{K}}_i$ does not invoke the knowledge extractor for the left **WIPoK-1**; Instead, it obtains all the preimages $\{x_i\}_{i\in[t]}$ for $\{y_i\}_{i\in[t]}$ in the left **Step-3** by *brute force*; It then picks a random index $s \xleftarrow{\$} [t]$ and uses preimage (s, x_s) as the witness to conduct the left **WIPoK-2**. First, observe that if the (s, x_s) picked by $\widehat{\mathcal{K}}_i$ hits the (j, x_j) extracted from the left **WIPoK-1** in $\mathcal{G}_i$ (see Fig. 3a), then the games $\widehat{\mathcal{K}}_i$ and $\mathcal{G}_i$ are identical (modulo that $\widehat{\mathcal{K}}_i$ is inefficient). Since $\widehat{\mathcal{K}}_i$ picks s uniformly at random from $[t]$, we know that $\widehat{\mathcal{K}}_i$ will be identical to $\mathcal{G}_i$ with probability $\geq 1/t$. It then follows from Inequality (1) that

$$\forall i \in [\widetilde{t}],\ \ \Pr\Big[w' = (i, \widetilde{x}_i) \text{ in } \widehat{\mathcal{K}}_i\Big] \geq \frac{1}{t}\cdot p - \mathsf{negl}(\lambda). \tag{2}$$

Next, observe that in $\widehat{\mathcal{K}}_i$, **WIPoK-1** is not rewound anymore. Therefore, by the (non-uniform, to compensate for the brute-forcing step) WI property of the right **WIPoK-1**, it follows that

$$\forall i \in [\widetilde{t}],\ \ \Big|\Pr\Big[w' = (i, \widetilde{x}_i) \text{ in } \widehat{\mathcal{K}}_i\Big] - \Pr\Big[w' = (i, \widetilde{x}_i) \text{ in } \widehat{\mathcal{K}}_1\Big]\Big| \leq \mathsf{negl}(\lambda). \tag{3}$$

Combining Inequalities (2) and (3), we have

$$\forall i \in [\widetilde{t}],\ \ \Pr\Big[w' = (i, \widetilde{x}_i) \text{ in } \widehat{\mathcal{K}}_1\Big] \geq \frac{1}{t}\cdot p - \mathsf{negl}(\lambda). \tag{4}$$

Inequality (4) implies the following

$$\Pr\Big[w' \text{ is a valid witness in } \widehat{\mathcal{K}}_1\Big] \geq \widetilde{t}\cdot\frac{1}{t}\cdot p - \mathsf{negl}(\lambda) \geq p + \frac{p}{t} - \mathsf{negl}(\lambda), \tag{5}$$

where "w' is a valid witness" refers to the event that $w' = (\widetilde{m}, \widetilde{r}) \vee w' = (1, \widetilde{x}_1) \vee \ldots \vee w' = (\widetilde{t}, \widetilde{x}_{\widetilde{t}})$, and the last "$\geq$" follows from the requirement that $t < \widetilde{t}$ (i.e., we are in the **One-Sided** setting).

On the other hand, we claim that

$$\Pr\Big[w' \text{ is a valid witness in } \widehat{\mathcal{K}}_1\Big] \leq p + \mathsf{negl}(\lambda). \tag{6}$$

This can be seen by comparing $\widehat{\mathcal{K}}_1$ with the real MIM execution shown in Fig. 1: Recall that p is the probability of R being convinced *in the real MIM execution*. The only difference between $\widehat{\mathcal{K}}_1$ and the real MIM execution is the witness used in the left **WIPoK-2**. By the (non-uniform) WI property of the left **WIPoK-2**, we know that R must be convinced in $\widehat{\mathcal{K}}_1$ with probability $\leq p + \mathsf{negl}(\lambda)$. Then, by the proof of knowledge property, we must extract a valid witness in the right **WIPoK-2** in $\widehat{\mathcal{K}}_1$ with probability upper-bounded by $p + \mathsf{negl}(\lambda)$ as well (this is exactly Inequality (6)).

Observe that Inequality (6) contradicts Inequality (5) because p/t is non-negligible. This gives us the desired contradiction, thus finishing the proof of Lem. 1.

Completing the Proof of Non-Malleability. With Lem. 1, non-malleability can be reduced to the computational-hiding property of Naor's commitment as follows. Assuming that $\mathcal{M}$ breaks the non-malleability of our scheme w.r.t. a distinguisher D, we construct an adversary $\mathcal{A}_{\mathsf{hiding}}$ against the computational-hiding property of Naor's commitment as follows:

[9] For readers trying to find a correspondence between this technical overview and its main-body counterpart (i.e., Sec. 4), we would like to point out that the $\widehat{\mathcal{K}}_i$ defined here does not match any machine in Sec. 4. In the current overview, $\widehat{\mathcal{K}}_i$ actually serves the functionality of several machines defined in Sec. 4.5 to 4.7. But $\widehat{\mathcal{K}}_i$ is most similar to the $\mathcal{K}''_i$ on Page 40 (and depicted in Fig. 9b), which is essentially $\widehat{\mathcal{K}}_i$ but additionally runs a knowledge extractor for the right **WIPoK-2**.

- **Main-Thread Simulation:** $\mathcal{A}_{\mathsf{hiding}}$ runs $\mathcal{G}_1$ where it embeds the instance of the hiding game of Naor's commitment in the left **Step-1** and **Step-2**, and finishes other steps just as $\mathcal{G}_1$. If R rejects, it sets $\widetilde{m} := \bot$. (Notice that at the end of this step, $\mathcal{M}$ will give an output that is computationally indistinguishable with the one from the real MIM execution. This is due to the similarity between $\mathcal{G}_1$ and the real MIM execution.)
- **Rewinding:** Unless R rejects in the above, $\mathcal{A}_{\mathsf{hiding}}$ repeats the following N (to be specified later) times to extract the message $\widetilde{m}$ committed in the right session:
  - It rewinds $\mathcal{M}$ to the point right after the completion of **Step-2**.
  - It picks a random index $i \xleftarrow{\$} [\widetilde{t}]$, and finishes the execution in the same manner as $\mathcal{G}_i$ (depicted in Fig. 3a).
  - Extract a witness $\widetilde{w}$ from the right **WIPoK-2** of the simulated execution.
  - If $\widetilde{w} = (\widetilde{m}, \widetilde{r})$ is a valid opening to $\widetilde{\mathsf{com}}$, it breaks the loop. Otherwise, it continues.

  If it fails to extract $\widetilde{m}$ within N trials, it aborts and outputs a random guess.
- **Decision:** It runs the distinguisher D on $\mathcal{M}$'s final output at the end of the **Main-Thread Simulation** step and $\widetilde{m}$, and outputs whatever D outputs.

It is easy to see that the above reduction works unless it fails to extract $\widetilde{m}$. We show that the failure probability can be made an arbitrarily small noticeable $\varepsilon(\lambda)$ if we take sufficiently large $N = \mathsf{poly}(\lambda)$ depending on ε. We call the snapshot (including the transcript and $\mathcal{M}$'s internal state) at the end of **Step-2** of the main thread a *prefix*. For each prefix pref, let p_{pref} be the probability that the right receiver accepts in $\mathcal{G}_1$ starting from pref. Then, Lem. 1 ensures that[10] for any fixed prefix pref such that $p_{\mathsf{pref}} \geq \varepsilon(\lambda)$, if we run $\mathcal{G}_i$ for random $i \xleftarrow{\$} [\widetilde{t}]$ and extract $\widetilde{w}$ from the right **WIPoK-2** (as is done in each repetition described in the above **Rewinding** step), then we will extract $\widetilde{w} = (\widetilde{m}, \widetilde{r})$ with at least another noticeable probability $\varepsilon'(\lambda)$ (More accurately, it should be the $\varepsilon'(\lambda)$ guaranteed by Lem. 1 *divided by* $\widetilde{t}$ as we pick an i randomly from $[\widetilde{t}]$. But this does not affect the argument here as $\widetilde{t}$ is bounded by a polynomial on λ—recall that we are in the **Small-Tag** setting.) Now, we consider the following two cases:

- The Case of $p_{\mathsf{pref}} \geq \varepsilon(\lambda)$: In this case, if we set $N = \Omega(\varepsilon'(\lambda)^{-1} \cdot \lambda)$, the reduction algorithm fails with at most a negligible probability (i.e., $(1 - \varepsilon'(\lambda))^{\Omega(\varepsilon'(\lambda)^{-1} \cdot \lambda)}$).
- The Case of $p_{\mathsf{pref}} < \varepsilon(\lambda)$: In this case, the right R rejects in the main-thread except for a probability upper-bounded by $\varepsilon(\lambda)$. Thus, the reduction fails with probability at most $\varepsilon(\lambda)$.

Overall, the reduction algorithm works with an additive loss of $\varepsilon(\lambda)$. Since we have the freedom to set $\varepsilon(\lambda)$ to an arbitrarily small noticeable function, the reduction from non-malleability to the computational hiding of Naor's commitment can be done properly. We omit further details.

Why Our Proof Is "Quantum-Friendly". Before going on, let us explain why the above proof of non-malleability in the classical setting could extend to the post-quantum setting. At a high-level, our security proof enjoys the following properties:

[10] Strictly speaking, there are two minor differences from Lem. 1. First, we are considering experiments for each fixed pref whereas Lem. 1 considers the whole experiment. This is not an issue since the proof of Lem. 1 does not touch messages before **Step-3** and thus works for any fixed pref. Another difference is that we define p_{pref} w.r.t. $\mathcal{G}_1$ whereas p is defined w.r.t. the real MIM experiment in Lem. 1. This is not an issue since $\mathcal{G}_1$ is computationally indistinguishable from the real MIM experiment.

– The rewinding-extraction procedure described on Page 12 never goes to touch the left Naor's commitment, which is the "hiding gadget" that provides computational hardness for the reduction. This property saves us from the robust simulatable extractability issue described in Sec. 1.1.

– Moreover, this extraction procedure is to first perform the "main-thread" execution as hybrid $\mathcal{G}_1$; If this main thread is accepted, then it starts N "rewinding threads" for extraction. This structure is very similar to the classical extractor for the canonical three-round extractable commitments in [PW09]. Thus, it allows us to invoke (a generalization of) the quantum simulatable-extraction lemma from [CCLY22] (developed originally to quantize the [PW09] extractable commitments) to quantize the our proof of non-malleability. More details are provided in Sec. 2.3.

In contrast, no previous constructions of (classical) constant-round non-malleable commitments achieved the above two properties *simultaneously*. This is why it is hard to quantize their security proofs *even with the [CCLY22] lemma in hand.*

On the Necessity of N Rewindings. Given that Lem. 1 already guarantees a noticeable probability $p'(\lambda)$ for successful extraction, one may wonder why we need to perform N rewindings in the above reduction to the computational hiding of Naor's commitment. That is, machine $\mathcal{A}_{\mathsf{hiding}}$ seems to work as desired even if we set $N = 1$—If it extracts $\widetilde{m}$, then the reduction is done; Otherwise, simply ask the reduction to guess at random in the Naor's hiding game. At the first glance, this seems to give a $\frac{1}{2} + p'(\lambda)$ advantage, which is sufficient for the security reduction.

Though this strategy looks appealing due to its simplicity, it does not work. To explain the reason, we need to refer to Lem. 9 on page Page 31 (i.e., the formal version of Lem. 1). The explanation (provided below) is rather technical and orthogonal to the current discussion. The reader may feel free to skip it and continue at Sec. 2.3 directly.

Notice that Lem. 9 refers to the prefix pref of the MIM execution, which consists of the first two rounds of the protocol. In particular, pref includes the left Naor's commitment from the hiding challenger. Lem. 9 says that for any noticeable probability ε, if a pref leads to an accepting MIM execution with probability greater than ε, then the extraction succeeds with a probability *lower-bounded* by $\varepsilon'/\widetilde{t}$, a value polynomially related to (but smaller than) ε.

At a high level, Lem. 9 is not directly applicable in our reduction because there may be a correlation between $\mathcal{M}$'s view and the success of extraction. That is, conditioned on the success of extraction, the *joint distribution* of $\mathcal{M}$'s view and the extracted right value may be distinguishable from those in the real MIM experiment. Consider the following distinguishing example: there are two prefixes pref_1 and pref_2 that lead to an accepting execution with the same probability ε, and they appear with the same probability in the real experiment. Suppose that the extraction success probability is $\varepsilon'/\widetilde{t}$ for pref_1, but ε $(> \varepsilon'/\widetilde{t})$ for pref_2. Then, if we run the reduction with Lem. 9 directly, the distinguisher may get to see pref_2 much more often than pref_1 (conditioned on successful extraction). But they should happen with the same probability in the real experiment.

The N rewindings in our construction essentially "amplify" Lem. 9 to the following stronger version: For any noticeable ε, if $\mathcal{M}$ convinces R with probability greater than ε, then the extraction succeeds with overwhelming probability (i.e., $1 - \mathsf{negl}(\lambda)$). This effectively ensures the following: Conditioned on a prefix that leads to acceptance with probability greater than ε *and* that R is indeed convinced, the joint distribution of $\mathcal{M}$'s view and the extracted $\widetilde{m}$ are negligibly close to their distribution in the real MIM execution. (Other cases are easy to handle: If R is not convinced, we simply use $\bot$ as the extracted value; Also, by an averaging argument, no more than ε fraction of prefixes can lead to acceptance with probability $< \varepsilon$.) Thus, our strategy eventually makes the joint distribution from simulation to be ε-close to the real one. Since we have the freedom to pick any noticeable ε, our reduction can achieve a non-negligible advantage in Naor's hiding game.

We refer the reader to Sec. 4.4 for a formal treatment of this issue.

2.3 Small-Tag, One-Sided, Synchronous, Post-Quantum Setting

Next, we explain how to make the $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{OneSided}}$ described in Sec. 2.1 non-malleable in the *post-quantum* setting.

For that, we simply instantiate $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{OneSided}}$ with post-quantum building blocks. (We denote this post-quantum version by $\langle C, R\rangle_{\mathsf{tg,PQ}}^{\mathsf{OneSided}}$.) Here, we have to be careful about what security notion we should require for WIPoK. In the security proof in the classical setting, we often extract a witness from WIPoK while continuing the rest of the experiment. For this to work, we implicitly require the knowledge extractor to have a *simulation* property [HSS11, LN11], i.e., it can simulate the prover's internal state as in a real execution, while performing the extraction task. While this is almost trivial in the classical setting, it becomes hard in the post-quantum setting due to the no-cloning theorem. In particular, constant-round constructions for such "simulatable" post-quantum WIPoKs are not known from (polynomially-hard) post-quantum OWFs.[11]

Fortunately, a recent work [CCLY22] shows that a constant-round construction from post-quantum OWFs is possible if we (i) relax the PoK to *argument* of knowledge (AoK), which only requires extractability against (quantum) polynomial-time adversaries, and (ii) relax the simulation requirement to the ε-*close* simulation property, where an (arbitrarily small) noticeable simulation error is allowed. Next, we show that such a ε-simulatable WIAoK suffices for our purpose. First, the relaxation from PoK to AoK is totally fine since we only apply the knowledge extractors for polynomial-time adversaries (possibly with non-uniform advice). Second, a noticeable error coming from extraction does not affect the proof of non-malleability if we take the noticeable error to be much smaller than the assumed MIM adversary's advantage. Thus, we ignore the noticeable errors, and assume that the error is negligible in the rest of this overview.

By using such a post-quantum WIAoK (as well as the post-quantum version of Naor's commitment Com and a post-quantum injective OWF f), we can see that the main technical part of the security proof in the classical setting (the proof of Lem. 1) can be migrated to the post-quantum setting almost immediately.

The only non-trivial issue is how to complete the reduction to the computational hiding of Naor's commitment, assuming (a post-quantum analog of) Lem. 1. In the classical setting, we rely on a rewinding argument that is not applicable anymore when $\mathcal{M}$ is a quantum machine (again, due to the no-cloning theorem). Thus, we use a different argument here—We observe that the following lemma, which is a generalization of [CCLY22, Lemma 4], suffices for this step.

Lemma 2 (Extract-and-Simulate Lemma (Informal)). *Let $\mathcal{G}$ and $\mathcal{K}$ be QPT algorithms that satisfy the following:*[12]

1. *$\mathcal{K}$ takes 1^λ and a quantum state ρ as input and outputs some unique s^* or otherwise $\perp$.*
2. *For any noticeable function $\gamma(\lambda)$, there exists a noticeable function $\delta(\lambda)$ such that for any polynomial-size quantum state ρ, if*

$$\Pr\Big[b = \top : (b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^\lambda, \rho)\Big] \geq \gamma(\lambda),$$

[11] A recent work by Lombardi, Ma, and Spooner [LMS21] gave such WIPoKs from super-polynomial hardness of post-quantum OWFs.

[12] In the formal version of this lemma (Lem. 20), $\mathcal{G}$ and $\mathcal{K}$ additionally take $1^{\gamma^{-1}}$ as input so that they can run extractors with simulation errors depending on γ. We ignore this in this overview since we assume negligibly-close simulation for simplicity.

then

$$\Pr\Big[\mathcal{K}(1^\lambda,\rho) = s^*\Big] \geq \delta(\lambda).$$

Then, there exists a QPT algorithm $\mathcal{SE}$ such that for any polynomial-size quantum state ρ and noticeable function $\varepsilon = \varepsilon(\lambda)$,

$$\{\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \rho)\}_\lambda \overset{s}{\approx}_\varepsilon \{(\rho_{\mathsf{out}}, \mathit{\Gamma}_b(s^*)) : (b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^\lambda, \rho)\}_\lambda \tag{7}$$

where

$$\mathit{\Gamma}_b(s^*) := \begin{cases} s^* & \textit{if } b = \top \\ \bot & \textit{otherwise} \end{cases}.$$

The above lemma is a generalization of [CCLY22, Lemma 4] and can be proven in a similar manner.[13] We will explain the main intuition for this proof at the end of this subsection.

Assuming a post-quantum analog of Lem. 1, we can complete the reduction to the computational hiding of Naor's commitment by using Lem. 2 as follows: We let ρ be the state of the experiment $\mathcal{G}_1$ (as defined in Fig. 3a, Sec. 2.1) right after finishing **Step-2**; Let $\mathcal{G}$ be the experiment that runs the rest of $\mathcal{G}_1$ starting from the state ρ;[14] At the end, $\mathcal{G}$ outputs the final state of $\mathcal{M}$ as ρ_{out}, and outputs the final decision of the right receiver R as b. We define $\mathcal{K}$ as follows:

- Upon receiving a state ρ, simulate $\mathcal{G}_i$ for $i \xleftarrow{\$} [\widetilde{t}]$.
- Extract a witness $\widetilde{w}$ from the right **WIPoK-2** of the simulated execution.
- If $\widetilde{w} = (\widetilde{m}, \widetilde{r})$ is a valid opening to $\widetilde{\mathsf{com}}$, output $\widetilde{m}$. Otherwise, output $\bot$.

It is worth noting that $\mathcal{K}$ corresponds to the repeated steps in the rewinding argument in Sec. 2.1. By construction, it is clear that $\mathcal{K}$ outputs $\widetilde{m}$ whenever it does not output $\bot$ (for an overwhelming fraction of $\widetilde{\beta}$ by the statistical binding property of Naor's commitment). Therefore, it satisfies Property 1 in Lem. 2 with $s^* := \widetilde{m}$ (for each fixed prefix). (We stress that we set $s^* = \widetilde{m}$ instead of $s^* = (\widetilde{m}, \widetilde{r})$ for ensuring the uniqueness of s^* by the statistical binding property of Naor's commitment.) Moreover, noting that $\mathcal{G}_1$ is computationally indistinguishable from the real MIM experiment, Lem. 1 implies that $\mathcal{K}$ also satisfies Property 2 in Lem. 2. Thus, there exists a machine $\mathcal{SE}$ that satisfies Eq. (7) for $s^* = \widetilde{m}$. Using this $\mathcal{SE}$, we can construct an adversary $\mathcal{A}_{\mathsf{hiding}}$ against the computational-hiding property of Naor's commitment as follows.

Assuming that $\mathcal{M}$ breaks the non-malleability of our scheme w.r.t. a distinguisher D, we construct an adversary $\mathcal{A}_{\mathsf{hiding}}$ against the computational-hiding property of Naor's commitment as follows.

- **Prefix Generation:** $\mathcal{A}_{\mathsf{hiding}}$ runs the real MIM experiment[15] for an adversary $\mathcal{M}$ until **Step-2** finishes, where it embeds the instance of the hiding game of Naor's commitment in the left **Step-2** while simulating other steps of the left C and the right R honestly. Let ρ be the state of the experiment at this point.
- **Extract-and-Simulate:** It runs $\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \rho)$ to extract $\widetilde{m}^*$ while simulating the output ρ_{out} of the experiment $\mathcal{G}_1$. We remark that it succeeds in extracting $\widetilde{m}^*$ whenever R accepts in the simulated experiment except for probability ε.

[13] Indeed, such a generalized version was mentioned in the technical overview of [CCLY22].

[14] Similar to the classical setting in Sec. 2.1, we stipulate that ρ does *not* include the left committer C's internal state ST_C at the end of **Step-2** (otherwise, we cannot reduce non-malleability to the hiding of the left commitment). This requirement is valid as the remaining steps of $\mathcal{G}_1$, which will be executed by $\mathcal{G}$, do not depend on ST_C.

[15] Remark that the real MIM experiment and $\mathcal{G}_1$ are identical until **Step-2**.

– **Decision:** It runs the distinguisher D on ρ_{out} and $\widetilde{m}$, and outputs whatever D outputs.

By Lem. 2, the $(\rho_{\mathsf{out}}, \widetilde{m})$ given to D is indistinguishable from that in $\mathcal{G}_1$ (up to an error ε), which in turn is indistinguishable from that of the real MIM experiment. Since we have the freedom to set ε to any arbitrarily small noticeable function, this completes the reduction.

Proof Idea of Lem. 2. We assume that ρ is a pure state w.l.o.g., and denote it by $|\psi\rangle$. By a similar usage of Jordan's lemma as in [CCY21, CCLY22], for any noticeable function $\delta(\lambda)$, we can decompose $|\psi\rangle$ as $|\psi\rangle = |\psi_{<\delta}\rangle + |\psi_{\geq\delta}\rangle$ such that

$$\Pr\Big[\mathcal{K}(1^\lambda, |\psi_{<\delta}\rangle) = s^*\Big] < \delta(\lambda) \tag{8}$$

and

$$\Pr\Big[\mathcal{K}(1^\lambda, |\psi_{\geq\delta}\rangle) = s^*\Big] \geq \delta(\lambda). \tag{9}$$

By the contraposition of Property 2 of Lem. 2 and Eq. (8), it holds that

$$\Pr\Big[b = \top : (b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^\lambda, |\psi_{<\delta}\rangle)\Big] < \gamma(\lambda),$$

where $\gamma(\lambda)$ is a noticeable function that can be made arbitrarily small by taking sufficiently small $\delta(\lambda)$. That is, $\mathcal{G}$ simply "rejects" (i.e., outputs $b = \bot$) except for a small probability $\gamma(\lambda)$, in which case $\mathit{\Gamma}_b(s^*) = \bot$. The simulation of this case is easy because we do not need to extract s^* except for probability $\gamma(\lambda)$, which can be made arbitrarily small. This shows Lem. 2 for the case where $|\psi\rangle$ only has $|\psi_{<\delta}\rangle$ component.

On the other hand, by Eq. (9), we can extract s^* from $|\psi_{\geq\delta}\rangle$ with an overwhelming probability if we can repeat $\mathcal{K}(1^\lambda, |\psi_{\geq\delta}\rangle)$ for $\mathit{\Theta}(\delta(\lambda)^{-1}\lambda)$ times. Though such a repetition is not possible in general, the way of defining $|\psi_{\geq\delta}\rangle$ (as explained in [CCLY22]) enables us to do so. As a result, we can extract s^* from $|\psi_{\geq\delta}\rangle$ almost without disturbing the state by the Almost-as-Good-as-New lemma [Aar05, Lemma 2.2] (a.k.a. the gentle measurement lemma), which shows Lem. 2 for the case where $|\psi\rangle$ only has $|\psi_{\geq\delta}\rangle$ component.

In the above, we separately analyze $|\psi_{<\delta}\rangle$ and $|\psi_{\geq\delta}\rangle$. In fact, we can see from the way we define them that they do not interfere with each other during the above described extraction procedure. As a result, the case of a superposition of $|\psi_{<\delta}\rangle$ and $|\psi_{\geq\delta}\rangle$ can be reduced to the above two cases.

See Lem. 20 for the formal statement of the lemma and Appx. A for the full proof.

2.4 Full-Fledged Post-Quantum Non-Malleable Commitments

To make $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$ a full-fledged post-quantum non-malleable commitments, we need to remove the one-side, small-tag, and synchronous restrictions. Due to space constraints, we will only briefly describe how to do that in this overview. We will provide more detailed overviews in the corresponding sections in the main body.

Removing the One-Sided Restriction. Notice that the non-malleability proof of $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ (and $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$) makes use of the one-sided condition in both **Step-3** and **WIPoK-1**. Let us refer to these two steps as a **Slot**. As demonstrated in Sec. 2.1 , it is the asymmetry condition $t < \widetilde{t}$ for this **Slot** that allows us to argue that the extracted w' must contain the right committed value $\widetilde{m}$ with good-enough probability.

We can use the following (standard) approach to remove the one-sided restriction. We simply repeat the **Slot** twice, sequentially. The first repetition, referred to as **Slot-A**, is identical to the original **Slot**; In particular, both parties use the tag t. The second repetition, referred to as **Slot-B**, will instead use $(n-t)$ as the tag.

Now, in a MIM execution, if $t < \widetilde{t}$, we can perform the same non-malleable proof as in Sec. 2.1 (and Sec. 2.3) with **Slot-A** playing the role of **Slot**. Otherwise (i.e., $t > \widetilde{t}$), it must hold that $(n-t) < (n-\widetilde{t})$; In this case, we can perform the non-malleable proof again, with **Slot-B** playing the role of **Slot**. (Note that by the definition of non-malleability, we do not need to consider the case $t = \widetilde{t}$.)

Handling Asynchronous Schedules. First, we observe that the proof strategy used in Sec. 2.1, which is for the synchronous setting, relies crucially on the following three conditions:

1. Both **Step-1** and **Step-2** in both the left and right sessions should finish before **Step-3** in either of the two sessions starts. This is because our security proof considers each fixed prefix, which is the state right after the end of **Step-2** (and before the start of **Step-3**).

2. The right **WIPoK-2** should not be interleaved with the left **WIPoK-1**. This is because $\mathcal{G}_i$ rewinds the left **WIPoK-1**; Meanwhile, we need to rewind the right **WIPoK-2** to extract $\widetilde{m}$. We do not want these two parts to rewind each other recursively.

3. The left **Step-3** should happen before the right **WIPoK-2**. This is because in machine $\widehat{\mathcal{K}}_i$, brute-forcing is performed for the left **Step-3**; At the same time, we need to rely on the WI property of the right **WIPoK-2** to argue the similarity among all the $\widehat{\mathcal{K}}_i$'s for different $i \in [\widetilde{t}]$ (i.e., Inequality (3)).

To prove non-malleability in the asynchronous setting, our strategy is to introduce more gadgets into the protocol that enforce the above three conditions. As long as these conditions are satisfied, we can then rely on essentially the same proof of non-malleability as in the synchronous case.

For example, to enforce the third condition above, we can add an extractable commitment ExtCom between **Step-3** and **WIPoK-1**, where C commits to m again (and additionally proves in **WIPoK-2** that he did it as required). Then, in the MIM execution, if the left **Step-3** happens after (the first message of) the right **WIPoK-2**, then the right ExtCom must happen before the left **Step-3**. In this case, we can extract $\widetilde{m}$ from the right ExtCom to finish the proof of non-malleability, instead of performing the same proof as in Sec. 2.1. We will use similar ideas to enforce all the conditions mentioned above.

Of course, in the real proof, we need to do the above argument for the two slot-version described above in the **Removing the One-Sided Restriction** part. Moreover, whenever we add one more gadget, cautions are needed to resolve new problems that it may cause by interacting with the "already-introduced" gadgets in some undesirable way. We will elaborate on this in Sec. 6, where we also provide a more detailed overview (in Sec. 6.1).

Tag Amplification. After the above two steps, we have already obtained a small-tag, asynchronous post-quantum non-malleable commitment $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg,PQ}}$. As mentioned earlier, now we can apply the [BLS22] tag amplifier. Here is one more caveat—The tag amplifier requires $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg,PQ}}$ to additionally satisfy a condition called *robust extractability*, which means that the extractability holds even if the cheating committer interacts with an external bounded-round machine that cannot be rewound. We show that our $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg,PQ}}$ already satisfies it, thanks to the added instances of ExtCom (which was introduced to handle asynchronous schedules originally). Thus, we can apply their tag amplifier to our $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg,PQ}}$ protocol to obtain the final full-fledged post-quantum non-malleable commitment scheme.

2.5 Organization

In Sec. 3, we present necessary notations and preliminaries.

In Sec. 4, we show the small-tag, one-sided, synchronous non-malleable commitment $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ in the classical setting.

In Sec. 5, we remove the one-sided restriction, obtaining the small-tag, synchronous non-malleable commitment $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$ (still in the classical setting).

In Sec. 6, we show how to handle asynchronous schedules *in the classical setting*. This leads to the small-tag, synchronous non-malleable commitment $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg}}$. As mentioned earlier, the way we handle asynchronous schedules is quantum-friendly. We could have done this part directly for *the quantum version of* $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$; But we find that it conveys our ideas better to first show the techniques in the classical setting, because it can avoid confusing the reader with quantum phenomena that are orthogonal to the discussion of asynchronous schedules. Once one understands our techniques for asynchronous schedules in the classical setting, it is easy to see why it extends to the post-quantum setting.

In Sec. 7, we show a post-quantum extract-and-simulate lemma, which plays an important role later when we quantize our classical proof of non-malleability.

In Sec. 8, we describe the full-fledged post-quantum non-malleable commitment. To do that, we first show in Sec. 8.1 how to quantize the non-malleable proof of $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$. This gives its post-quantum analog $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$ (i.e., a small-tag, one-sided, synchronous, post-quantum non-malleable commitment). Next, we describe in Sec. 8.2 how to remove the one-sided and synchronous restrictions from $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$, to obtain the protocol $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$; This step mimics the techniques we used in the classical setting converting $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ to $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg}}$. Finally, in Sec. 8.3, we obtain the final full-fledged post-quantum non-malleable commitment by applying the [BLS22] tag amplifier to $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$.

In Sec. 9, we show the application of our post-quantum commitments to quantum-secure MPC.

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.

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. An adversary (or malicious party) is modeled as a non-uniform PPT (or QPT in the case of post-quantum security) algorithm. When we consider a non-uniform QPT adversary, we often 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 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 $\{\mathcal{A}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$.

For an **NP** language $\mathcal{L}$ and a true statement in this language $x \in \mathcal{L}$, we use $\mathcal{R}_\mathcal{L}(x)$ ($\mathcal{R}$ stands for "relation") to denote the set of all witnesses for x. We will refer to the OR-composition of **NP** languages, which are defined in Def. 1.

Definition 1 (OR-Composition of NP Languages). *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two* **NP** *languages. The* OR-composition *of them (dubbed $\mathcal{L}_1 \vee \mathcal{L}_2$) is the new* **NP** *language defined as follows:*

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

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 $\mathcal{X} \overset{c}{\approx}_\delta \mathcal{Y}$ 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

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

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 $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$.

Similarly, we use $\mathcal{X} \overset{s}{\approx}_\delta \mathcal{Y}$ to mean that for any unbounded time algorithm $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $i \in I_\lambda$, we have

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

In particular, when the above hold for $\delta = 0$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are statistically indistinguishable, and simply write $\mathcal{X} \overset{s}{\approx} \mathcal{Y}$. Moreover, we write $\mathcal{X} \equiv \mathcal{Y}$ 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 Commitment Schemes

We define (classically-secure and post-quantum) commitments. The following definitions are based on those in [CCLY22].

Definition 2 (Commitment). *A* commitment scheme *is a pair of PPT ITMs $\langle C, R\rangle$ that satisfies the following. 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 τ, a decommitment* decom *as C's private output, and a decision value $b_{\mathrm{com}} \in \{\top, \bot\}$ as R's private output. We denote this execution by $(\tau, \mathsf{decom}, b_{\mathrm{com}}) \leftarrow \langle C(m), R\rangle(1^\lambda)$.*
- **Decommit Stage:** *C sends the committed message m and the decommitment* decom *to R, and R outputs decision $b_{\mathrm{dec}} \in \{\top, \bot\}$. We denote this execution by $b_{\mathrm{dec}} \leftarrow \mathsf{Verify}(\tau, m, \mathsf{decom})$.*[16] *W.l.o.g., we assume that whenever R rejects at the end of the commit stage (i.e., $b_{\mathrm{com}} = \bot$), R will reject at the end of decommit stage (i.e., $b_{\mathrm{dec}} = \bot$). (Note that w.l.o.g., one can think that b_{com} is included in τ, because we can always modify the protocol to ask R to send b_{com} as the last-round message of the commit stage.)*

[16] We assume that there is no state information for R kept from the commit stage.

The scheme satisfies the following completeness requirement:

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

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

Definition 3 (Computationally Hiding). *A commitment scheme* $\langle C, R\rangle$ *is* computationally hiding *if for any non-uniform PPT 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.*

We say that $\langle C, R\rangle$ *is* post-quantum *computationally hiding if the above holds for all non-uniform* QPT R^*.

Definition 4 (Statistically Binding). *A commitment scheme* $\langle C, R\rangle$ *is* statistically binding *if for any unbounded-time committer* C^*, *the following holds:*

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

Definition 5 (Committed Values). *For a commitment scheme* $\langle C, R\rangle$, *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 τ *and* decom *are defined in the commit stage in Def. 2.*

3.3 Non-Malleable Commitments

Classically Non-Malleable Commitments. We first define non-malleable commitments in the classical setting. This definition follows the formalization in [LPV08, GPR16]. We consider a man-in-the-middle adversary $\mathcal{M}$ interacting with a committer C in the *left*, and a receiver R in the *right*. We denote the relevant entities used in the right interaction as the "tilde'd" version of the corresponding entities on the left. In particular, suppose that C commits to m in the left interaction, and $\mathcal{M}$ commits to $\widetilde{m}$ on the right, i.e., we set $\widetilde{m} = \mathsf{val}(\widetilde{\tau})$ where $\widetilde{\tau}$ is the transcript of the right session. Let $\mathsf{mim}^{\mathcal{M}}_{\langle C,R\rangle}(\lambda, m, z)$ denote the random variable that is the pair $(\mathsf{OUT}_{\mathcal{M}}, \widetilde{m})$, consisting of $\mathcal{M}$'s output as well as the value committed to by $\mathcal{M}$ on the right (assuming C commits to m on the left), where z is $\mathcal{M}$'s non-uniform advice. We use a *tag-based* (or "identity-based") specification, and ensure that $\mathcal{M}$ uses a distinct tag $\widetilde{t}$ on the right from the tag t it uses on the left. This is done by stipulating that $\mathsf{mim}^{\mathcal{M}}_{\langle C,R\rangle}(\lambda, m, z)$ outputs a special value $\bot_{tag}$ when $\mathcal{M}$ uses the same tag in both the left and right executions. The reasoning is that this corresponds to the uninteresting case when $\mathcal{M}$ is simply acting as a channel, forwarding messages from C on the left to R on the right and vice versa.

Definition 6 (Classical Non-Malleable Commitments). *A commitment scheme* $\langle C, R\rangle$ *is said to be* (classically) non-malleable *if for every (non-uniform) PPT man-in-the-middle adversary* $\mathcal{M}$ *and every polynomial* $\ell : \mathbb{N} \to \mathbb{N}$, *it holds that*

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

Post-Quantum Non-Malleable Commitments. In the post-quantum setting, non-malleability can be defined similarly, except that the man-in-the-middle adversary can be QPT, instead of being PPT. (In particular, the final output of the adversary could be a quantum state.) To avoid using confusing notations, we put an over-line on top of the variable mim in the post-quantum setting.

Definition 7 (Post-Quantum Non-Malleable Commitments). *A commitment scheme $\langle C, R\rangle$ is said to be* post-quantumly non-malleable *if for every (non-uniform) QPT man-in-the-middle adversary $\mathcal{M} = \{\mathcal{M}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ and every polynomial $\ell : \mathbb{N} \to \mathbb{N}$, it holds that*

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

where "$\overset{c}{\approx}$" refers to computational indistinguishability against QPT distinguishers.

Remark 1 (On Entangled Auxiliary Information.). The above definition 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.

Synchronous Adversaries: This notion refers to man-in-the-middle adversaries who upon receiving a message in the left (reps. right) session, immediately respond with the corresponding message in the right (resp. left) session. An adversary is said to be *asynchronous* if it is not synchronous.

3.4 Extractable Commitments

Classically Extractable Commitments. A commitment scheme is extractable if there exists an efficient extractor such that, the committed value can be extracted. We present the definition in Def. 8, which is taken from [PW09]. Constant-round constructions of (classically) extractable commitments are known from OWFs [PW09].

Definition 8 (Classically Extractable Commitments). *A commitment* $\mathsf{ExtCom} = \langle C, R\rangle$ *is* extractable *if there exists an expected polynomial-time probabilistic oracle machine (the extractor) $\mathcal{E}$ that given oracle access to any PPT cheating committer C^* outputs a pair (τ, σ^*) such that:*

- **Simulation:** *τ is identically distributed to the view of C^* at the end of interacting with an honest receiver R in commitment phase.*
- **Extraction:** *the probability that τ is accepting and $\sigma^* = \bot$ is negligible.*
- **Binding:** *if $\sigma^* \neq \bot$, then it is statistically impossible to open τ to any value other than σ^*.*

Post-Quantum (Robust) Extractable Commitments. We define the post-quantum analog of extractable commitments. In the post-quantum setting, we often need an extractor that (almost) does not disturb the (potentially malicious) committer's state during the extraction. However, it is not known that such a post-quantum extractable commitments exist from (polynomially hard) post-quantum OWFs.[17] Fortunately, a recent work [CCLY22] 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 [BLS22] with some notational adaptations.

[17] A recent work of [LMS21] gave a constant-round construction of such extractable commitments from super-polynomial hardness of post-quantum OWFs.

Let $\langle C, R\rangle$ be a (possibly tag-based) commitment scheme. A sequential committed-value oracle $O^\infty[\langle C, R\rangle]$ acts as follows in interaction with a committer C^*: it interacts with C^* in many sequential sessions; in each session,

– it participates with C^* in the commit phase of $\langle C, R\rangle$ as the honest receiver R (using a tag chosen adaptively by C^*), obtaining a transcript τ, **and**

– if C^* is non-aborting in the commit phase and sends request break, it returns $\mathsf{val}(\tau)$.

The single-session oracle $O^1[\langle C, R\rangle]$ is similar to $O^\infty[\langle C, R\rangle]$, except that it interacts with the adversary in a single session. When the commitment scheme is clear from the context, we write O^∞ (or O^1) for simplicity.

Then, the definition of post-quantum ε-simulatable commitment is given below.

Definition 9 (Post-Quantum ε-Simulatable Extractable Commitment). *A commitment scheme $\langle C, R\rangle$ is said to be a* post-quantum ε-simulatable extractable commitment *if it satisfies the following in addition to post-quantum computational hiding and statistical binding. There exists a QPT algorithm $\mathcal{SE}$ such that for any noticeable $\varepsilon(\lambda)$ and any non-uniform QPT committer $\{C^*_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$,*

$$\left\{\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, C^*_\lambda, \rho_\lambda)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx}_\varepsilon \left\{C^{*O^\infty}_\lambda(\rho_\lambda)\right\}_{\lambda\in\mathbb{N}},$$

where "$\overset{c}{\approx}_\varepsilon$" refers to ε-close computational indistinguishability against QPT distinguishers.

Lemma 3 ([CCLY22, Lemma 10]). *Assuming the existence of post-quantum OWFs, there exist constant-round post-quantum ε-simulatable extractable commitments.*

Remark 2. The ε-simulatable extractability as defined in Def. 9 is called post-quantum *strong* ε-simulatable extractability in [CCLY22]. Though their definition looks different from Def. 9, they are actually equivalent—First, it is easy to see that Def. 9 is equivalent to a modified version of Def. 9 where we give O^1 instead of O^∞. Second, [BLS22, Lemma 3.2] shows that the O^1 and O^∞ versions of Def. 9 are equivalent.

We also define the *robust* version of the above definition following [BLS22]. Roughly speaking, r-robust ε-simulatable extractability means that the ε-simulatable extractability holds even if the adversary interacts with an external r-round interactive machine that cannot be rewound by the extractor. The formal definition is given below.

Definition 10 (Post-Quantum r-Robust ε-Simulatable Extractable Commitment). *A commitment scheme $\langle C, R\rangle$ is said to be a* post-quantum r-robust ε-simulatable extractable commitment *if it satisfies the following in addition to post-quantum computational hiding and statistical binding. There exists a QPT algorithm $\mathcal{SE}$ such that for any noticeable $\varepsilon(\lambda)$, non-uniform QPT committer $\{C^*_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$, and r-round interactive machine B,*

$$\left\{\mathsf{OUT}_{\mathcal{SE}}\langle B(1^\lambda, z), \mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, C^*_\lambda, \rho_\lambda)\rangle\right\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^*} \overset{c}{\approx}_\varepsilon \left\{\mathsf{OUT}_{C^*_\lambda}\langle B(1^\lambda, z), C^{*O^\infty}_\lambda(\rho_\lambda)\rangle\right\}_{\lambda\in\mathbb{N}, z\in\{0,1\}^*}$$

where "$\overset{c}{\approx}_\varepsilon$" refers to ε-close computational indistinguishability against QPT distinguishers.

3.5 Witness-Indistinguishable Proofs of Knowledge

Classically Witness-Indistinguishable Proofs of Knowledge. First, we provide a definition from [Gol01, Lin03].

Definition 11 (Proofs of Knowledge [Gol01, Lin03]). *Let $\mathcal{L}$ be an* **NP** *language and $\kappa : \mathbb{N} \rightarrow [0,1]$. We say that a pair of PPT ITMs $\langle P, V\rangle$ is an interactive proof of knowledge for $\mathcal{L}$ with knowledge error κ if the following two conditions hold:*

1. **Completeness.** *For every true statement $x \in \mathcal{L}$ and every witness $w \in \mathcal{R}_{\mathcal{L}}(x)$, it holds that*
$$\Pr[\mathsf{OUT}_V\langle P(w), V\rangle(x) = \top] = 1.$$
2. **Validity (with error κ).** *There exists a polynomial $q(\cdot)$ and a probabilistic oracle machine K such that for every P^*, and every $x, z, r \in \{0,1\}^*$, machine K satisfies the following condition:*
   Denote by $p(x,z,r)$ the probability that $V(x)$ accepts when interacting with the prover specified by $P^*(x,z;r)$. If $p(x,z,r) > \kappa(|x|)$, then, on input x and with oracle access to oracle $P^*(x,z;r)$, K outputs a solution $w \in \mathcal{R}_{\mathcal{L}}(x)$ within an expected number of steps bounded by $\frac{q(|x|)}{p(x,z,r)-\kappa(|x|)}$.

We present an alternative formalism of knowledge soundness called *witness-extended emulation* due to Lindell [Lin03]. This formalism turns out to be more suitable for our application.

Definition 12 (PoK via Witness-Extended Emulation [Lin03]). *Let $\mathcal{L}$ be an* **NP** *language and $\kappa : \mathbb{N} \rightarrow [0,1]$. We say that a pair of ITMs $\langle P, V\rangle$ is an interactive proof of knowledge for $\mathcal{L}$ with* witness-extended emulation *if it satisfies the same completeness requirement as in Def. 11, and the following requirement: There exists an expected PPT oracle machine* WE*, called the* emulation-extractor*, such that for any machine P^* the following conditions hold*

1. $\{(\mathsf{ST}'_{P^*}, b') : (\mathsf{ST}'_{P^*}, b', w') \leftarrow \mathcal{WE}^{P^*(x,z;r)}(x)\}_{\lambda,x,z,r} \equiv \{(\mathsf{ST}_{P^*}, b) : (\mathsf{ST}_{P^*}, b) \leftarrow \langle P^*(x,z;r), V(x)\rangle\}_{\lambda,x,z,r}$*, where in the RHS above, ST_{P^*} and b denote the state of P^* and output of V respectively, at the end of the execution $\langle P^*(x,z;r), V(x)\rangle$.*
2. $\Pr\Big[b' = 1 \ \wedge \ w' \notin \mathcal{R}_{\mathcal{L}}(x) : (\mathsf{ST}'_{P^*}, b', w') \leftarrow \mathcal{WE}^{P^*(x,z;r)}(x)\Big] = \mathsf{negl}(|x|)$*.*

The following lemma connects Def. 12 with Def. 11.

Lemma 4 ([Lin03, Lemma 3.1]). *Let $\langle P, V\rangle$ be a proof of knowledge for $\mathcal{L} \in$* **NP** *with negligible knowledge error. Then, $\langle P, V\rangle$ is also a proof of knowledge for $\mathcal{L}$ with witness-extended emulation.*

Next, we define witness indistinguishable proof of knowledge:

Definition 13 (WIPoKs). *A witness-indistinguishable proof of knowledge for an* **NP** *language $\mathcal{L}$ is a proof of knowledge for $\mathcal{L}$ with witness extended emulation (as per Def. 12) which additionally satisfies the following property*

– **Witness-Indistinguishability.** *For any non-uniform PPT machine V^*, we have*
$$\{\mathsf{OUT}_{V^*}\langle P(w_0), V^*\rangle(x)\}_{\lambda,x,w_0,w_1} \overset{c}{\approx} \{\mathsf{OUT}_{V^*}\langle P(w_1), V^*\rangle(x)\}_{\lambda,x,w_0,w_1},$$
where $\lambda \in \mathbb{N}$, $x \in \mathcal{L} \cap \{0,1\}^\lambda$, and $w_0, w_1 \in \mathcal{R}_{\mathcal{L}}(x)$.

It is well-known that constant-round witness-indistinguishable proofs of knowledge (as per Def. 13) can be built from one-way functions.

Post-Quantum Witness-Indistinguishable Arguments of Knowledge We define a post-quantum analog of the WIPoKs described in Def. 13. A natural way to do so is to just replace all adversaries with any QPT algorithms. More precisely, we require that the witness indistinguishability holds against QPT distinguishers and that the knowledge extractor extracts a witness without disturbing the prover's state. However, a constant-round construction for such WIPoKs is not known based on (polynomially hard) post-quantum OWFs.[18] Fortunately, a recent work [CCLY22] showed that a constant-round construction from post-quantum OWFs is possible if we relax the proof of knowledge via witness-extended emulation to *argument* of knowledge via witness-extended *ε-close* emulation, where a cheating prover is limited to QPT and a (arbitrarily small) noticeable simulation error is allowed. A formal definition of such post-quantum WIAoKs is given below.

Definition 14 (Post-Quantum WIAoK with ε-Close Emulation). *Let $\mathcal{L}$ be an* **NP** *language. We say that a pair of PPT ITMs $\langle P, V\rangle$ is a post-quantum witness-distinguishable arguments of knowledge with ε-close emulation for $\mathcal{L}$ if the following three conditions hold:*

1. **Completeness.** *For every $(x, w) \in \mathcal{L}$, it holds that*

$$\Pr[\mathsf{OUT}_V\langle P(w), V\rangle(x) = \top] = 1.$$

2. **AoK via Witness-Extended ε-Close Emulation.** *There exists a QPT oracle machine $\mathcal{WE}$, called the* witness-extended emulator, *such that for any non-uniform QPT machine $\{P^*_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ the following conditions hold: for any noticeable function $\varepsilon(\cdot)$ (referred to as the error parameter),*

   (a) $\{(\mathsf{ST}'_{\mathcal{P}^*}, b') : (\mathsf{ST}'_{\mathcal{P}^*}, b', w') \leftarrow \mathcal{WE}(1^{\varepsilon^{-1}}, P^*_\lambda, \rho_\lambda, x)\}_{\lambda,x} \overset{c}{\approx}_\varepsilon \{(\mathsf{ST}_{\mathcal{P}^*}, b) : (\mathsf{ST}_{P^*}, b) \leftarrow \langle P^*_\lambda(\rho_\lambda), V(x)\rangle\}_{\lambda,x}$ *where $\lambda \in \mathbb{N}$, $x \in \{0,1\}^\lambda$, and "$\overset{c}{\approx}_\varepsilon$" refers to ε-computational indistinguishability against QPT distinguishers;*

   (b) $\Pr\Big[b' = 1 \ \wedge \ w' \notin \mathcal{R}_\mathcal{L}(x) : (\mathsf{ST}'_{\mathcal{P}^*}, b', w') \leftarrow \mathcal{WE}(1^{\varepsilon^{-1}}, P^*_\lambda, \rho_\lambda, x)\Big] = 0,$[19]

   where $(\mathsf{ST}_{P^*}, b) \leftarrow \langle P^*_\lambda(\rho^*_\lambda), V(x)\rangle$ means that ST_{P^*} is the final state of P^*_λ and b is the decision output by V.

3. **Witness-Indistinguishability.** *For any non-uniform QPT machine $\{V^*_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$, we have*

$$\{\mathsf{OUT}_{V^*_\lambda}\langle P(w_0), V^*_\lambda(\rho_\lambda)\rangle(x)\}_{\lambda,x,w_0,w_1} \overset{c}{\approx} \{\mathsf{OUT}_{V^*_\lambda}\langle P(w_1), V^*_\lambda(\rho_\lambda)\rangle(x)\}_{\lambda,x,w_0,w_1},$$

   where $\lambda \in \mathbb{N}$, $x \in \mathcal{L} \cap \{0,1\}^\lambda$, $w_0, w_1 \in \mathcal{R}_\mathcal{L}(x)$, and "$\overset{c}{\approx}$" refers to computational indistinguishability against QPT distinguishers.

The work of [CCLY22] gives a constant-round construction of post-quantum *ε-zero-knowledge* arguments of knowledge from post-quantum OWFs. Since ε-zero-knowledge implies witness indistinguishability, we have the following lemma.

Lemma 5 ([CCLY22, Corollary 2]). *Assuming the existence of post-quantum OWFs, there exist constant-round post-quantum witness-indistinguishable arguments of knowledge as per Def. 14.*

4 Small-Tag, One-Sided, Synchronous, Classical Setting

In this section, we show the small-tag, one-sided, synchronous non-malleable commitment scheme $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ in the classical setting. Though our final goal is to construct *post-quantum* non-malleable commitments, we give a classically-secure construction here since the security proof for its post-quantum version $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg,PQ}}$ (in Appx. B) is very similar to that in the classical case except for one step where we rely on the extract-and-simulate lemma (Lem. 20).

[18] A recent work of [LMS21] gave such WIPoKs from super-polynomial hardness of post-quantum OWFs.

[19] We can assume the probability is 0 instead of $\mathsf{negl}(|x|)$ without loss of generality. If the probability is $\mathsf{negl}(|x|)$, we can modify $\mathcal{WE}$ to output $b' = 0$ whenever $w' \notin \mathcal{R}_\mathcal{L}(x)$. This only negligibly affects the distribution in the LHS of Condition 2a, which can be absorbed into $\overset{c}{\approx}_\varepsilon$.

4.1 Construction of $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{OneSided}}$

An intuitive explanation of this construction is already provided in Sec. 2.1. We now present the formal description in Prot. 1. This construction is based on the following building blocks:

– An *injective* OWF f; To convey the main idea better, we first present the construction assuming the existence of *injective* OWFs. We explain how to relax the assumption to the existence to *any* OWFs in Sec. 4.8.

– Naor's commitment; We use β to denote the first message for Naor's commitment, and $\mathsf{Com}_\beta(m;r)$ to denote the second message where a string m is committed using randomness r.

– A witness-indistinguishable proof of knowledge WIPoK.

Protocol 1: Small-Tag One-Sided Synchronous NMCom $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{OneSided}}$

The tag space is defined to be $[n]$ where n is a polynomial on λ. Let $t \in [n]$ be the tag for the following interaction. Let m be the message to be committed to.

Commit Stage:

1. Receiver R samples and sends the first message β for Naor's commitment;
2. Committer C commits to m using the second message of Naor's commitment. Formally, C samples a random tape r and sends $\mathsf{com} = \mathsf{Com}_\beta(m;r)$;
3. R computes $\{y_i = f(x_i)\}_{i\in[t]}$ with $x_i \xleftarrow{\$} \{0,1\}^\lambda$ for each $i \in [t]$. R sends $Y = (y_1, \ldots, y_t)$ to C;
4. **(WIPoK-1.)**[a] R and C execute an instance of WIPoK where R proves to C that he "knows" the pre-image of some y_i contained in Y (defined in Step 3). Formally, R proves that $Y \in \mathcal{L}_f^t$, where
$$\mathcal{L}_f^t \coloneqq \{(y_1, \ldots, y_t) \mid \exists (i, x_i) \ s.t.\ i \in [t] \wedge y_i = f(x_i)\}. \tag{10}$$
Note that R uses $(1, x_1)$ as the witness when executing this WIPoK.
5. **(WIPoK-2.)** C and R execute an instance of WIPoK where C proves to R that he "knows" *either* the message committed in com (defined in Step 2), *or* the pre-image of some y_i contained in Y (defined in Step 3). Formally, C proves that $(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}_f^t$, where $\mathcal{L}_\beta \vee \mathcal{L}_f^t$ denotes the OR-composed language (as per Def. 1), $\mathcal{L}_f^t$ was defined in Language (10) and
$$\mathcal{L}_\beta \coloneqq \{\mathsf{com} \mid \exists (m, r) \ s.t.\ \mathsf{com} = \mathsf{Com}_\beta(m;r)\}. \tag{11}$$
Note that C uses the (m, r) defined in Step 2 as the witness when executing this WIPoK.

Decommit Stage: C sends (m, r). R accepts if $\mathsf{com} = \mathsf{Com}_\beta(m;r)$, and rejects otherwise.

[a] We assume that the first round of this step goes to the opposite direction to Step 3. This is true for typical WIPoK constructions. See Rmk. 3 for more details.

Remark 3 (Inserting an ACK round if necessary). In Prot. 1, we emphasize that Step 3 and the first message of Step 4 should be *separated*, i.e., they cannot be combined into one round even if the first message of Step 4 happens to go from R to C as well, which may be possible depending on the choice of WIPoK protocols instantiating Step 4. In that case, one can simply insert a dummy round right after Step 3, where C sends an "ACK" symbol to acknowledge the reception of Step 3; Only then will R starts the execution of Step 4. This is necessary because some steps in our security proof need to non-uniformly fix the execution up to the closing of Step 3, and make use of the (non-uniform) WI property of Step 4.

Security. Completeness is straightforward from the description of Prot. 1. The statistical binding property follows from that of Naor's commitment. Computational-hiding property of any non-malleable commitment scheme follows directly from its non-malleability. So, we only need to prove the non-malleability of our protocol, which is established by the following theorem.

Theorem 2. *The commitment scheme $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ in Prot. 1 is non-malleable against one-sided synchronous PPT adversaries with tag space $[n]$, with n being any polynomial on λ.*

We prove Thm. 2 in subsequent subsections.

Structure of the Proof of Thm. 2. Since the proof of Thm. 2 is lengthy, we provide a road map here. We prove Thm. 2 in Sec. 4.2 to 4.7. In each subsection, we introduce a lemma or claim whose proof is deferred to the next subsection. Specifically,

- In the rest of Sec. 4.2, we prove Thm. 2 assuming that Lem. 6 is correct.
- In Sec. 4.3, we prove Lem. 6 assuming that Lem. 7 is correct.
- In Sec. 4.4, we prove Lem. 7 assuming that Lem. 9 is correct.
- In Sec. 4.5, we prove Lem. 9 assuming that Lem. 10 is correct.
- In Sec. 4.6, we prove Lem. 10 assuming that Claim 10 is correct.
- In Sec. 4.7, we prove Claim 10.

Putting everything together, we complete the proof of Thm. 2.

Remark 4. We remark that the proof of Lem. 7 is the only step that cannot be directly translated to the post-quantum setting. Looking ahead, the post-quantum counterpart of this step (i.e., Lem. 28) will make use of a new *post-quantum extract-and-simulate* lemma, which we state and prove in Sec. 7.

Moreover, we provide a moderately-detailed summary of this proof in Sec. 4.9, with the hope to further elucidate the logic flow behind the proof of Thm. 2. It is recommended that the reader start reading Sec. 4.9 after he/she has read Sec. 4.2 to 4.8 (at least once), because Sec. 4.9 uses several notations defined in preceding subsections.

4.2 Proving Non-Malleability (Proof of Thm. 2)

In the rest of this section, we write $\langle C, R\rangle$ to mean $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ for notational simplicity. To prove Thm. 2, let us first define the game that captures the man-in-the-middle execution corresponding to $\mathsf{mim}^{\mathcal{M}}_{\langle C,R\rangle}(\lambda, m, z)$ w.r.t. the $\langle C, R\rangle$ defined in Prot. 1.

Game $H^{\mathcal{M}}(\lambda, m, z)$: This is the man-in-the-middle execution of the commit stage of the $\langle C, R\rangle$ defined in Prot. 1, where the left committer commits to m and $\mathcal{M}$'s non-uniform advice is z. The output of this game consists of the following three parts:

1. $\mathsf{OUT}_{\mathcal{M}}$: this is the output of $\mathcal{M}$ at the end of this game;
2. $\widetilde{\tau}$: this is defined to be $\widetilde{\tau} := (\widetilde{\beta}, \widetilde{\mathsf{com}})$, where $\widetilde{\beta}$ and $\widetilde{\mathsf{com}}$ are the Step 1 and Step 2 messages exchanged between $\mathcal{M}$ and the honest receiver R (i.e., in the right session);
3. $b \in \{\top, \bot\}$: this is the output of the honest receiver R, indicating if the man-in-the-middle's commitment (i.e., the right session) is accepted ($b = \top$) or not ($b = \bot$).

Notation. For any $(\mathsf{OUT}_{\mathcal{M}}, \widetilde{\tau}, b)$ in the support of $H^{\mathcal{M}}(\lambda, m, z)$, the following $\mathsf{val}_b(\widetilde{\tau})$ defines the value committed in the man-in-the-middle's commitment (i.e., the right session):

$$\mathsf{val}_b(\widetilde{\tau}) := \begin{cases} \mathsf{val}(\widetilde{\tau}) & b = \top \\ \bot & b = \bot \end{cases}, \tag{12}$$

where $\mathsf{val}(\widetilde{\tau})$ denote the value statistically-bound in Steps 1 and 2 of the right session. (Recall that these two steps constitute a Naor's commitment.) Also, throughout this paper, we assume for simplicity that Naor's commitment is perfectly binding (instead of only statistically binding). This assumption only suppresses an *additive error of negligible amount* in relevant lemmas, which will not affect any of our results.

By definition, it is easy to see that:

$$\left\{\mathsf{mim}^{\mathcal{M}}_{\langle C,R\rangle}(\lambda, m, z)\right\} \overset{\text{i.d.}}{=\!=} \left\{\big(\mathsf{OUT}_{\mathcal{M}}, \mathsf{val}_b(\widetilde{\tau})\big) : (\mathsf{OUT}_{\mathcal{M}}, \widetilde{\tau}, b) \leftarrow H^{\mathcal{M}}(\lambda, m, z)\right\}, \tag{13}$$

where both ensembles are indexed by $\lambda \in \mathbb{N}$, $m \in \{0,1\}^{\ell(\lambda)}$, and $z \in \{0,1\}^*$.

Therefore, to prove that Prot. 1 satisfies Def. 6, it suffices to show the following:

$$\begin{aligned} &\left\{\big(\mathsf{OUT}^0_{\mathcal{M}}, \mathsf{val}_{b^0}(\widetilde{\tau}^0)\big) : (\mathsf{OUT}^0_{\mathcal{M}}, \widetilde{\tau}^0, b^0) \leftarrow H^{\mathcal{M}}(\lambda, m_0, z)\right\} \\ \overset{\text{c}}{\approx}\ &\left\{\big(\mathsf{OUT}^1_{\mathcal{M}}, \mathsf{val}_{b^1}(\widetilde{\tau}^1)\big) : (\mathsf{OUT}^1_{\mathcal{M}}, \widetilde{\tau}^1, b^1) \leftarrow H^{\mathcal{M}}(\lambda, m_1, z)\right\}, \end{aligned} \tag{14}$$

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

Proof by Contradiction. Our high-level strategy is proof by contradiction. We assume for contradiction that there are a (possibly non-uniform) PPT machine $\mathcal{D}$ and a function $\delta(\lambda) = 1/\mathsf{poly}(\lambda)$ such that for infinitely many $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[\mathcal{D}\big(1^\lambda, \mathsf{OUT}^0_{\mathcal{M}}, \mathsf{val}_{b^0}(\widetilde{\tau}^0)\big) = 1\right] - \Pr\left[\mathcal{D}\big(1^\lambda, \mathsf{OUT}^1_{\mathcal{M}}, \mathsf{val}_{b^1}(\widetilde{\tau}^1)\big) = 1\right]\right| \geq 3 \cdot \delta(\lambda), \tag{15}$$

where the first probability is taken over the random procedure $(\mathsf{OUT}^0_{\mathcal{M}}, \widetilde{\tau}^0, b^0) \leftarrow H^{\mathcal{M}}(\lambda, m_0, z)$ and the second probability is taken over the random procedure $(\mathsf{OUT}^1_{\mathcal{M}}, \widetilde{\tau}^1, b^1) \leftarrow H^{\mathcal{M}}(\lambda, m_1, z)$.

We then show the following lemma:

Lemma 6. *There exists a hybrid G such that for any PPT $\mathcal{M}$ and the $\delta(\lambda)$ defined above, the following holds*

1. $\left\{(\mathsf{OUT}^0, \mathsf{Val}^0) : (\mathsf{OUT}^0, \mathsf{Val}^0) \leftarrow G^{\mathcal{M}}(\lambda, m_0, z)\right\} \overset{\text{c}}{\approx} \left\{(\mathsf{OUT}^1, \mathsf{Val}^1) : (\mathsf{OUT}^1, \mathsf{Val}^1) \leftarrow G^{\mathcal{M}}(\lambda, m_1, z)\right\}$, *where both ensembles are indexed by $\lambda \in \mathbb{N}$, $(m_0, m_1) \in \{0,1\}^{\ell(\lambda)} \times \{0,1\}^{\ell(\lambda)}$, and $z \in \{0,1\}^*$.*
2. $\left\{(\mathsf{OUT}^G, \mathsf{Val}^G) : (\mathsf{OUT}^G, \mathsf{Val}^G) \leftarrow G^{\mathcal{M}}(\lambda, m, z)\right\} \overset{\text{c}}{\approx}_{\delta(\lambda)} \left\{(\mathsf{OUT}^H, \mathsf{val}_{b^H}(\widetilde{\tau}^H)) : (\mathsf{OUT}^H, \widetilde{\tau}^H, b^H) \leftarrow H^{\mathcal{M}}(\lambda, m, z)\right\}$, *where both ensembles are indexed by $\lambda \in \mathbb{N}$, $m \in \{0,1\}^{\ell(\lambda)}$, and $z \in \{0,1\}^*$.*

It is easy to see that if Lem. 6 is true, it contradicts our assumption in Inequality (15). Thus, it will finish the proof of non-malleability. Indeed, this lemma is the most technically involved part. We prove it in Sec. 4.3.

4.3 Proof of Lem. 6

First, we provide a new but equivalent interpretation of the game $H^{\mathcal{M}}(\lambda, m, z)$ in Algo. 4.1,. (We also provide a picture in Fig. 4a to illustrate it.)

Algorithm 4.1: Re-interpretation of Game $H^{\mathcal{M}}(\lambda, m, z)$

Game $H^{\mathcal{M}}(\lambda, m, z)$ can be split into the following stages:

1. **Prefix Generation:** First, execute Steps 1 and 2 of the man-in-the-middle game of Prot. 1. That is, it plays as the left honest committer committing to m and the right honest receiver, with $\mathcal{M}(1^\lambda, z)$ being the man-in-the-middle adversary.

   Notation: Let $\mathsf{st}_{\mathcal{M}}$ denote the state of $\mathcal{M}$ at the end of Step 2; Let st_C (resp. st_R) denote the state of the honest committer (resp. receiver) at the end of Step 2; Let $\widetilde{\tau} = (\widetilde{\beta}, \widetilde{\mathsf{com}})$[a] and $\tau = (\beta, \mathsf{com})$. In terms of notation, we denote the execution of this stage by

   $$(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau}) \leftarrow H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z). \tag{16}$$

   We will call the tuple $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ the *prefix* and denote it by pref. It is worth noting that this pref contains all the information such that a PPT machine can "complete" the remaining execution of $H^{\mathcal{M}}(\lambda, m, z)$ starting from pref.

2. **The Remainder:** Next, it simply resumes from where the **Prefix Generation** stage stops, to finish the remaining steps of the man-in-the-middle execution $H^{\mathcal{M}}(\lambda, m, z)$.

   Notation: We introduce the following notations to describe this stage. Define a PPT machine $\mathcal{A}$ that takes as input $(\mathsf{st}_{\mathcal{M}}, \widetilde{\tau})$; Machine $\mathcal{A}$ is supposed to run the residual strategy of $\mathcal{M}$ starting from $\mathsf{st}_{\mathcal{M}}$. Also, define a PPT machine $\mathcal{B}$ that takes as input $(\mathsf{st}_C, \mathsf{st}_R, \widetilde{\tau})$; Machine $\mathcal{B}$ is supposed to run the residual strategies of the honest committer C and receiver R, starting from st_C and st_R respectively[b]. With the above notations, we can denote the execution of the remaining steps of $H^{\mathcal{M}}(\lambda, m, z)$ by

   $$(\mathsf{OUT}_{\mathcal{A}}, b) \leftarrow \langle \mathcal{A}(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau}), \tag{17}$$

   where $\mathsf{OUT}_{\mathcal{A}}$ is the output of $\mathcal{A}$, and $b \in \{\bot, \top\}$ is the output of the honest receiver R (in the right), indicating if the man-in-the-middle's commitment (i.e., the right session) is accepted ($b = \top$) or not ($b = \bot$). (We remark that $\mathsf{OUT}_{\mathcal{A}}$ is nothing but the man-in-the-middle $\mathcal{M}$'s final output.)

3. **Output:** It outputs the tuple $(\mathsf{OUT}_{\mathcal{A}}, \widetilde{\tau}, b)$.

[a] Recall that $\widetilde{\beta}$ and $\widetilde{\mathsf{com}}$ are the Steps 1 and 2 messages in the right session; They constitute an execution of Naor's commitment.

[b] Note that it is actually redundant to give $\widetilde{\tau}$ as common input to these parties—It can be included in $\mathsf{st}_{\mathcal{M}}$ and st_R. We choose to make $\widetilde{\tau}$ explicit for notational convenience.

We claim the following lemma regarding Algo. 4.1.

Lemma 7. *Let $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$, $\mathcal{A}$, $\mathcal{B}$ be as defined in Algo. 4.1. There exists a PPT machine $\mathcal{SE}$ (the simulation-extractor) such that for any $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ in the support of $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$, any noticeable $\varepsilon(\lambda)$, it holds that*

$$\begin{aligned}&\left\{(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}) : (\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}}\\ \overset{c}{\approx}_{\varepsilon(\lambda)} &\left\{\left(\mathsf{OUT}_{\mathcal{A}}, \mathsf{val}_b(\widetilde{\tau})\right) : (\mathsf{OUT}_{\mathcal{A}}, b) \leftarrow \langle \mathcal{A}(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}}\end{aligned}$$

Remark 5. The third output $b_{\mathcal{SE}}$ of $\mathcal{SE}$ is redundant in the above lemma. We include it in the output of $\mathcal{SE}$ for convenience in the proof of this lemma in Sec. 4.4.

Lem. 7 is the main technical lemma for the current proof of Lem. 6. We present its proof in Sec. 4.4. In the following, we finish the proof of Lem. 6 assuming that Lem. 7 is true.

Now, we are ready to present the description of G as required by Lem. 6.

Algorithm 4.2: Hybrid $G^{\mathcal{M}}(\lambda, m, z)$

This hybrid proceeds as follows:

1. **Prefix Generation:** This stage is identical to Stage 1 of $H^{\mathcal{M}}(\lambda, m, z)$. Formally, it executes

$$(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau}) \leftarrow H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z),$$

where $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$ is defined in Stage 1 of Algo. 4.1.

2. **The Remainder:** Define $\mathcal{A}$ in the same way as in Stage 2 of $H^{\mathcal{M}}(\lambda, m, z)$. With this $\mathcal{A}$ and the $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ from the previous stage, $G^{\mathcal{M}}(\lambda, m, z)$ invokes the $\mathcal{SE}$ prescribed in Lem. 7. Formally, it executes the following procedure:

$$(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\delta^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau}),$$

where the δ is the statistical distance that we want to show for Property 2 of Lem. 6.

Remark 6. We emphasize that in this stage, $G^{\mathcal{M}}(\lambda, m, z)$ does *not* make use of st_C.

3. **Output:** It outputs $(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}})$.[a]

[a] Note that $b_{\mathcal{SE}}$ is not included in the output of G. Indeed, $b_{\mathcal{SE}}$ is not important for the current machine $G^{\mathcal{M}}(\lambda, m, z)$. We choose to include it in the output of $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\delta^{-1}}, \mathsf{st}_R, \widetilde{\tau})$ only for notational convenience when we define/prove properties about $\mathcal{SE}$ itself.

Proving Property 1 of Lem. 6. Observe that hybrid $G^{\mathcal{M}}(\lambda, m, z)$ is an efficient machine because both $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$ and $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\delta^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ are efficient. Moreover, it does not rewind Steps 1 and 2 of the man-in-the-middle execution, and $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\delta^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ does *not* need to know st_C. Therefore, Property 1 of Lem. 6 follows immediately from the computational-hiding property of the left Naor's commitment (i.e., Steps 1 and 2 in the left session).

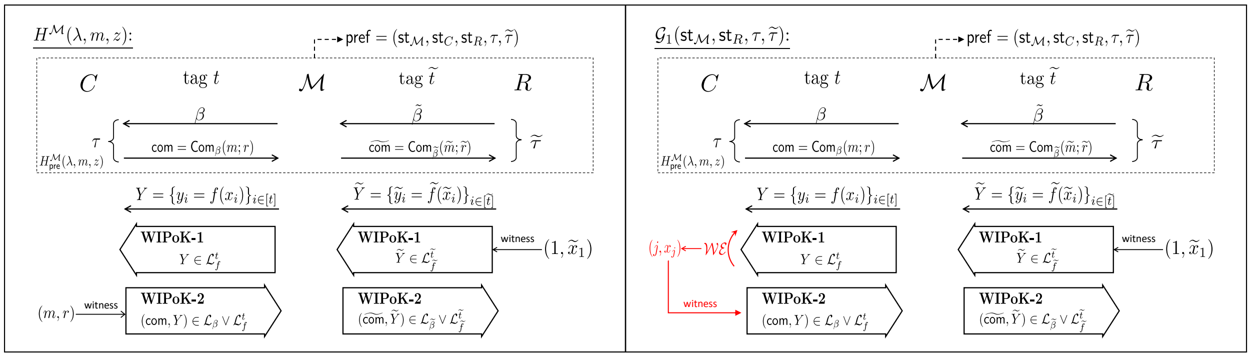
Proving Property 2 of Lem. 6. First, observe that the distribution of the prefix $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ is identical in $G^{\mathcal{M}}(\lambda, m, z)$ and $H^{\mathcal{M}}(\lambda, m, z)$. For each fixed prefix, Lem. 7 implies that $G^{\mathcal{M}}(\lambda, m, z)$ and $H^{\mathcal{M}}(\lambda, m, z)$ are $\delta(\lambda)$-computationally indistinguishable where we remark that $G^{\mathcal{M}}(\lambda, m, z)$ runs $\mathcal{SE}$ with the second input $1^{\delta^{-1}}$. This immediately implies Property 2 of Lem. 6.

4.4 Proof of Lem. 7

Now, we prove Lem. 7 by constructing the required machine $\mathcal{SE}$. Our machine $\mathcal{SE}$ (for Lem. 7) will make use of two machines: (1) a $\mathcal{G}_1$ that simulates the main-thread, and (2) an extractor $\mathcal{K}$ that extracts the value committed in $\widetilde{\tau}$. Thus, in the following, we will first define these two machines, and then present the description of $\mathcal{SE}$ in Algo. 4.3.

First, we describe a machine $\mathcal{G}_1$ that simulates the real execution *without using* st_C.

Machine $\mathcal{G}_1$: (Illustrated in Fig. 4b) For any prefix pref, $\mathcal{G}_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to Stage 2 of $H^{\mathcal{M}}(\lambda, m, z)$ shown in Algo. 4.1 (and depicted in Fig. 4a), except for the following difference: Instead of executing the left **WIPoK-1** honestly, it uses the witness-extended emulator $\mathcal{WE}$ (as per Def. 12) of the left **WIPoK-1** to extract a witness, and



$H^{\mathcal{M}}(\lambda, m, z)$:
pref = $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\tilde{\beta}$
τ
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\tilde{\beta}}(\widetilde{m}; \widetilde{r})$
$\widetilde{\tau}$
$H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$
$Y = \{y_i = f(x_i)\}_{i \in [t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i \in [\widetilde{t}]}$
WIPoK-1
$Y \in \mathcal{L}^t_f$
$\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
witness
$(1, \widetilde{x}_1)$
WIPoK-2
(m, r)
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\tilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
$\mathcal{G}_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
$(j, x_j) \leftarrow \mathcal{WE}$

(a) (b)

Fig. 4: Machines $H^{\mathcal{M}}$ and $\mathcal{G}_1$ (Difference is highlighted in red color)

– If the left committer accepts the left **WIPoK-1** and the extracted witness is valid (i.e., it is a (j, x_j) pair such that $y_j = f(x_j)$ for some $j \in [t]$), $\mathcal{G}_1$ uses (j, x_j) to finish the left **WIPoK-2**. Similarly to $H^{\mathcal{M}}(\lambda, m, z)$, $\mathcal{G}_1$ eventually outputs $\mathcal{M}$'s final state and the right receiver's decision bit b;

– If the left committer accepts the left **WIPoK-1** but the extracted witness is invalid, it aborts immediately and outputs $(\bot, \bot)$.

– If the left committer rejects the left **WIPoK-1**, it runs the rest of execution of $H^{\mathcal{M}}(\lambda, m, z)$ to output $\mathcal{M}$'s final state and the right receiver's decision bit b. Note that it does not need to run the left **WIPoK-2** in this case since the left committer aborts after the left **WIPoK-1**.

We denote the above procedure by $(\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$. It is worth noting that $\mathcal{G}_1$ does *not* need to know st_C.

Remark 7 (Precise Meaning of Extraction). In the above description of $\mathcal{G}_1$, we say that "it uses the witness-extended emulator of the left **WIPoK-1** to extract a witness". More accurately, this means the following: We consider a cheating prover against the left **WIPoK-1** that takes as advice the states of $\mathcal{M}$ and the right receiver right before the start of the left **WIPoK-1**, interacts with the left committer by simulating $\mathcal{M}$ and the right receiver, and outputs the states of $\mathcal{M}$ and the right receiver at the end of the left **WIPoK-1**. Machine $\mathcal{G}_1$ runs the witness-extended emulator of the left **WIPoK-1** (as per Def. 12) w.r.t. the this cheating prover to simulate the states of $\mathcal{M}$ and the right receiver right at the end of the left **WIPoK-1**, the decision bit of the left **WIPoK-1** of the left committer, while extracting a (candidate of) witness of $\mathcal{L}^t_f$. We will use similar convention many times throughout this paper.

Next, we prove a lemma that shows that $\mathcal{G}_1$ simulates the real MIM execution.

Lemma 8. *For the $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$, $\mathcal{A}$, and $\mathcal{B}$ defined in Algo. 4.1, for any* $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ *in the support of $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$, it holds that*

$$\big\{(\mathsf{OUT},\ b) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda \in \mathbb{N}}$$
$$\overset{c}{\approx} \big\{(\mathsf{OUT}_{\mathcal{A}},\ b) : (\mathsf{OUT}_{\mathcal{A}}, b) \leftarrow \langle \mathcal{A}(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})\big\}_{\lambda \in \mathbb{N}}.$$

Proof. We first define a hybrid machine $\mathcal{G}'_1$ below.

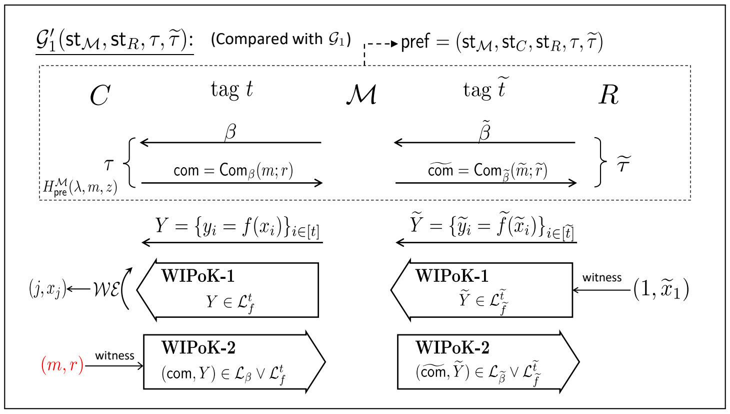

$\mathcal{G}'_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
(Compared with $\mathcal{G}_1$)
$\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\widetilde{\beta}$
τ
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
$\widetilde{\tau}$
$H^\mathcal{M}_\mathsf{pre}(\lambda, m, z)$
$Y = \{y_i = f(x_i)\}_{i\in[t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i\in[\widetilde{t}]}$
$(j, x_j) \leftarrow \mathcal{WE}$
WIPoK-1
$Y \in \mathcal{L}^t_f$
WIPoK-1
$\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
witness
$(1, \widetilde{x}_1)$
(m, r)
witness
WIPoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$
WIPoK-2
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$

Fig. 5: Machine $\mathcal{G}'_1$ (Difference with $\mathcal{G}_1$ is highlighted in red color)

Machine $\mathcal{G}'_1$: (Illustrated in Fig. 5) For any prefix pref, $\mathcal{G}'_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it uses (m, r) as the witness in the left **WIPoK-2** even if it succeeds in extracting a valid witness from the left **WIPoK-1**.

By the WI property of the left **WIPoK-2**, it holds that

$$\begin{aligned}&\big\{(\mathsf{OUT},\ b) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}}\\ \overset{\text{c}}{\approx}\ &\big\{(\mathsf{OUT},\ b) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}'_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}}\end{aligned}. \tag{18}$$

Note that the only difference between $\mathcal{G}'_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ and $\langle \mathcal{A}(\mathsf{st}_\mathcal{M}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})$ is that the former runs the witness-extended emulator of the left **WIPoK-1** (but does not use the extracted witness at all). Thus, by the PoK property of the left **WIPoK-1**, it holds that

$$\begin{aligned}&\big\{(\mathsf{OUT},\ b) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}'_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}}\\ \overset{\text{c}}{\approx}\ &\big\{(\mathsf{OUT}_\mathcal{A},\ b) : (\mathsf{OUT}_\mathcal{A}, b) \leftarrow \langle \mathcal{A}(\mathsf{st}_\mathcal{M}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}}\end{aligned}. \tag{19}$$

By combining Eq. (18) and (19), we obtain Lem. 8.

□

Next, we define the probability of R being convinced in the execution of $\mathcal{G}_1$. This value plays an important role later in our proof.

Definition 15. *For any* $\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ *in the support of* $H^\mathcal{M}_\mathsf{pre}(\lambda, m, z)$*, we define the following value* $p^\mathsf{Sim}_\mathsf{pref}$*:*

$$p^\mathsf{Sim}_\mathsf{pref} := \Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})].$$

Next, we show a technical lemma that gives an extractor $\mathcal{K}$ *without the simulation property.*

Lemma 9. *Let* $H^\mathcal{M}_\mathsf{pre}(\lambda, m, z)$*,* $\mathcal{A}$*, and* $\mathcal{B}$ *be as defined in Algo. 4.1. There exists an expected PPT machine* $\mathcal{K}$ *such that for any* $\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ *in the support of* $H^\mathcal{M}_\mathsf{pre}(\lambda, m, z)$ *and any noticeable* $\varepsilon(\lambda)$*, the following holds:*

1. **(Syntax.)** *$\mathcal{K}$ takes as input $(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$ and makes oracle access to $\mathcal{A}(\mathsf{st}_\mathcal{M})$. It outputs a value $\mathsf{Val}_\mathcal{K} \in \{0,1\}^{\ell(\lambda)} \cup \{\bot\}$ such that $\mathsf{Val}_\mathcal{K} = \mathsf{val}(\widetilde{\tau})$ whenever $\mathsf{Val}_\mathcal{K} \neq \bot$. (Also see Rmk. 8 for an intuitive explanation.)*

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

$$\Pr\Big[\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val}_{\mathcal{K}} \leftarrow \mathcal{K}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})\Big] \geq \frac{\varepsilon'(\lambda)}{\widetilde{t}},$$

where $\varepsilon'(\lambda) := \frac{\varepsilon(\lambda)}{10t^2}$.

Remark 8 (On the Output of $\mathcal{K}$). The output $\mathsf{Val}_{\mathcal{K}}$ is expected to be the value committed in $\widetilde{\tau}$, i.e., $\mathsf{val}(\widetilde{\tau})$. If $\mathsf{Val}_{\mathcal{K}} = \bot$, it indicates that $\mathcal{K}$ did not extract the correct $\mathsf{val}(\widetilde{\tau})$. We explicitly include $\bot$ in the range of $\mathcal{K}$ for the following purpose: Looking ahead, the simulation-extractor $\mathcal{SE}$ that we are going to build will invoke $\mathcal{K}$ for several times, until the value $\mathsf{val}(\widetilde{\tau})$ is extracted. However, $\mathcal{SE}$ cannot tell if the extracted value is indeed $\mathsf{val}(\widetilde{\tau})$. Thus, the case $\mathsf{Val}_{\mathcal{K}} = \bot$ serves as an indicator, telling $\mathcal{SE}$ if the extraction by $\mathcal{K}$ succeeds (more accurately, fails).

As a vigilant reader may have already realized, there is an alternative formalism: simply ask $\mathcal{K}$ to output both $\mathsf{val}(\tau)$ *and the decommitment information* so that $\mathcal{SE}$ can test by itself whether $\mathcal{K}$'s extraction is successful. We remark that this approach does work for the current proof *in classical setting*. However, it may not extend when we prove *post-quantum* non-malleability. In short, that is because to make this proof work in the quantum setting, our technique requires that the valid output of $\mathcal{K}$ should be *unique*; Only in this way can we ask $\mathcal{K}$ to "forget" other information, such that the extraction procedure can be "un-computed" to rewind $\mathcal{M}$ back without much disturbance to its initial state. (This point will become clearer in the proof of Lem. 20.) However, if we include the decommitment information in the output of $\mathcal{K}$, then the valid output may not be unique, even if the commitment scheme is perfect-binding—There could exists different ways to decommit to the *unique* committed value.

To make the proof consistent in both classical and quantum settings, we choose the current formalism in Property 1.

We will prove Lem. 9 in Sec. 4.5. In the rest of this subsection, we finish the proof of Lem. 7 assuming Lem. 9 is true.

We present the description of $\mathcal{SE}$ (for Lem. 7) in Algo. 4.3.

Algorithm 4.3: Simulation-Extractor $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$

Let $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$, $\mathcal{A}$, and $\mathcal{B}$ be as defined in Algo. 4.1. Let $\varepsilon(\lambda)$ be a noticeable function. For $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ in the support of $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$, machine $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ proceeds as follows:

1. **Main-Thread Simulation.** It first uses the machine $\mathcal{G}_1$ to simulate Stage 2 of $H^{\mathcal{M}}(\lambda, m, z)$. That is, it computes $(\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$. It sets $\mathsf{OUT}_{\mathcal{SE}} := \mathsf{OUT}$ and $b_{\mathcal{SE}} := b$. Then,
   - if $b_{\mathcal{SE}} = \bot$, it sets $\mathsf{Val}_{\mathcal{SE}} := \bot$ and jumps directly to Stage 4;
   - otherwise, it goes to the next step.
2. **Rewinding:** $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ then loops the following procedure for $\frac{\widetilde{t}}{\varepsilon'(\lambda)} \cdot \lambda$ times, where $\varepsilon'(\lambda) := \frac{\varepsilon(\lambda)}{10t^2}$:
   - Execute $\mathsf{Val}_{\mathcal{K}} \leftarrow \mathcal{K}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$, where $\mathcal{K}$ is provided by Lem. 9. If $\mathsf{Val}_{\mathcal{K}} \neq \bot$,[a] set $\mathsf{Val}_{\mathcal{SE}} := \mathsf{Val}_{\mathcal{K}}$ and jump to Stage 4; Otherwise, go to the next loop.
3. It sets $\mathsf{Val}_{\mathcal{SE}} = \bot$. (Note that if this stage is reached, it means that the number of the loop in the previous stage reached its upper bound $\frac{\widetilde{t}}{\varepsilon'(\lambda)} \cdot \lambda$, but $\mathcal{K}$ did not extract the message committed in $\widetilde{\tau}$ yet.)

4. **Output:** it outputs $(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}})$.

[a] Note that by Property 1 in Lem. 9, this means $\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau})$.

We first prove that the simulated main-thread is computationally indistinguishable from the real one.

Claim 3. *Let $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$, $\mathcal{A}$, $\mathcal{B}$ be as defined in Algo. 4.1. For any $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ in the support of $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$ and any noticeable $\varepsilon(\lambda)$, it holds that*

$$\begin{aligned}&\left\{\left(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau})\right) : (\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}} \\ \overset{c}{\approx}\ &\left\{\left(\mathsf{OUT}_{\mathcal{A}}, \mathsf{val}_{b}(\widetilde{\tau})\right) : (\mathsf{OUT}_{\mathcal{A}}, b) \leftarrow \langle\mathcal{A}(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^{\lambda}, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}}. \end{aligned} \tag{20}$$

Proof. Since we consider each fixed prefix, we can efficiently compute $\mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau})$ and $\mathsf{val}_b(\widetilde{\tau})$ from $b_{\mathcal{SE}}$ and b, respectively, by using $\mathsf{val}(\widetilde{\tau})$ as a non-uniform advice[20]. Then, it suffices to prove

$$\begin{aligned}&\left\{\left(\mathsf{OUT}_{\mathcal{SE}}, b_{\mathcal{SE}}\right) : (\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}} \\ \overset{c}{\approx}\ &\left\{\left(\mathsf{OUT}_{\mathcal{A}}, b\right) : (\mathsf{OUT}_{\mathcal{A}}, b) \leftarrow \langle\mathcal{A}(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^{\lambda}, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}}. \end{aligned}$$

Since $\mathcal{SE}$ just runs $\mathcal{G}_1$ in the main thread, the above follows directly from Lem. 8.

□

Given Claim 3, to prove Lem. 7, it suffices to show the following inequality:

$$\Pr\left[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) : (\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})\right] \leq \varepsilon(\lambda) + \mathsf{negl}(\lambda). \tag{21}$$

To prove Inequality (21), let us first define two events:

- **Event $E_{\leq\varepsilon}$:**[21] This is the event that the prefix $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ lead to an accepting execution of $\mathcal{SE}$ with probability at most ε. Formally, it denotes the event that $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ satisfies the following inequality:

$$\Pr\left[b_{\mathcal{SE}} = \top : (\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})\right] \leq \varepsilon(\lambda).$$

  Remark 9. We emphasize that the above probability (on the LHS) is exactly the $p^{\mathsf{Sim}}_{\mathsf{pref}}$ defined in Def. 15, because $\mathcal{SE}$ obtains $b_{\mathcal{SE}}$ by running $\mathcal{G}_1$ (see Stage 1 of Algo. 4.3).

- **Event E_{acc}:** This is the event that $b_{\mathcal{SE}} = \top$ at the end of the execution of $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$. Namely, it means that the honest receiver R accepts in the execution simulated by $\mathcal{SE}$. (The subscript "acc" stands for "accept").

We first show that the the probability of $(E_{\mathsf{acc}} \wedge E_{\leq\varepsilon})$ is upper-bounded by $\varepsilon(\lambda)$.

Claim 4. *It holds that*

$$\Pr\left[E_{\mathsf{acc}} \wedge E_{\leq\varepsilon} : (\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})\right] \leq \varepsilon(\lambda).$$

[20] Note that this is possible because both the LHS and the RHS of Eq. (20) use the same *fixed* $\widetilde{\tau}$, which does not depend on the randomness of the ensembles.

[21] We remark that this event actually does not depend on the random procedure $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^{\lambda}, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$. It is only about the property of the prefix $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ given as input to machine $\mathcal{SE}$.

Proof. This is an immediate result of the definitions of these two events. In more details,

$$\Pr[E_{\mathsf{acc}} \wedge E_{\leq\varepsilon}] = \Pr[E_{\mathsf{acc}} \mid E_{\leq\varepsilon}] \cdot \Pr[E_{\leq\varepsilon}] \leq \Pr[E_{\mathsf{acc}} \mid E_{\leq\varepsilon}] \cdot 1 = \varepsilon(\lambda) \cdot 1, \tag{22}$$

where all the probabilities are taken over $(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$.

□

Next, we upper-bound the LHS of Inequality (21) by $\varepsilon(\lambda) + \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \mid E_{\mathsf{acc}} \wedge \neg E_{\leq\varepsilon}]$:

Claim 5. *It holds that*

$$\Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau})] \leq \varepsilon(\lambda) + \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \mid E_{\mathsf{acc}} \wedge \neg E_{\leq\varepsilon}],$$

where both probabilities are taken over $(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$.

Proof. All the probabilities below are taken over $(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$:

$$\begin{aligned}
\Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau})] &= \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \wedge E_{\mathsf{acc}}] + \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \wedge \neg E_{\mathsf{acc}}] \\
&= \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \wedge E_{\mathsf{acc}}] \qquad (23) \\
&= \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \wedge E_{\mathsf{acc}} \wedge E_{\leq\varepsilon}] + \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \wedge E_{\mathsf{acc}} \wedge \neg E_{\leq\varepsilon}] \\
&\leq \Pr[E_{\mathsf{acc}} \wedge E_{\leq\varepsilon}] + \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \wedge E_{\mathsf{acc}} \wedge \neg E_{\leq\varepsilon}] \\
&\leq \varepsilon(\lambda) + \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \wedge E_{\mathsf{acc}} \wedge \neg E_{\leq\varepsilon}] \qquad (24) \\
&\leq \varepsilon(\lambda) + \Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \mid E_{\mathsf{acc}} \wedge \neg E_{\leq\varepsilon}],
\end{aligned}$$

where Eq. (23) follows from the fact that if $b_{\mathcal{SE}} = \bot$ then $\mathsf{Val}_{\mathcal{SE}} = \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) = \bot$, and Inequality (24) follows from Claim 4.

□

Recall that our goal is to establish Inequality (21). Due to Claim 5, it now suffices to show

$$\Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \mid E_{\mathsf{acc}} \wedge \neg E_{\leq\varepsilon}] = \mathsf{negl}(\lambda), \tag{25}$$

where the probability is taken over $(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$.

Proof of Eq. (25). Given E_{acc} (i.e., $b_{\mathcal{SE}} = \top$), we know that $\mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ must have entered Stage 2 to execute the machine $\mathcal{K}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$ for at most $\frac{\widetilde{t}}{\varepsilon'(\lambda)} \cdot \lambda$ times, with early termination only if the event $\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau})$ happens (recall that $\mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) = \mathsf{val}(\widetilde{\tau})$ when $b_{\mathcal{SE}} = \top$).

Given $\neg E_{\leq\varepsilon}$ (thus, $p^{\mathsf{Sim}}_{\mathsf{pref}} > \varepsilon(\lambda)$, see Rmk. 9), it follows from Property 2 in Lem. 9 that each time $\mathcal{K}$ is invoked, it outputs $\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau})$ with probability $\geq \varepsilon'(\lambda)/\widetilde{t}$.

Therefore, we have

$$\Pr[\mathsf{Val}_{\mathcal{SE}} \neq \mathsf{val}_{b_{\mathcal{SE}}}(\widetilde{\tau}) \mid E_{\mathsf{acc}} \wedge \neg E_{\leq\varepsilon}] = \left(1 - \frac{\varepsilon'(\lambda)}{\widetilde{t}}\right)^{\frac{\widetilde{t}}{\varepsilon'(\lambda)} \cdot \lambda} = 2^{-O(\lambda)} = \mathsf{negl}(\lambda),$$

where the probability is taken over $(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}, b_{\mathcal{SE}}) \leftarrow \mathcal{SE}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, 1^{\varepsilon^{-1}}, \mathsf{st}_R, \tau, \widetilde{\tau})$.

This finishes the proof of Eq. (25).

□

This eventually finishes the proof of Lem. 7 (modulo the proof of Lem. 9, which we present in Sec. 4.5).

4.5 Extractor $\mathcal{K}$ (Proof of Lem. 9)

Machine $\mathcal{G}_i$ $(i \in [\widetilde{t}])$: (Illustrated in Fig. 6.) Recall that we have already defined the machine $\mathcal{G}_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ on Page 29. Now, for $i \in [\widetilde{t}] \setminus \{1\}$, $\mathcal{G}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{G}_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it uses $(i, \widetilde{x}_i)$ as the witness in the right **WIPoK-1**.

Claim 6. $\forall i \in [\widetilde{t}],\ \Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda)$.

Proof. We define hybrid machines $\mathcal{G}'_i$ and $\mathcal{G}''_i$ as follows.

Machine $\mathcal{G}'_i$ $(i \in [\widetilde{t}])$: (Illustrated in Fig. 7a.) Recall that we have already defined the machine $\mathcal{G}'_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ on Page 31. Now, for $i \in [\widetilde{t}] \setminus \{1\}$, $\mathcal{G}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{G}'_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it uses $(i, \widetilde{x}_i)$ as the witness in the right **WIPoK-1**.

Machine $\mathcal{G}''_i$ $(i \in [\widetilde{t}])$: (Illustrated in Fig. 7b) For any prefix pref, $\mathcal{G}''_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{G}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it honestly runs the left **WIPoK-1**, instead of running the witness-extended emulator $\mathcal{WE}$. In other words, $\mathcal{G}''_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\langle \mathcal{A}(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})$ except that it uses $(i, \widetilde{x}_i)$ as the witness in the right **WIPoK-1**. In particular, $\mathcal{G}''_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ is identical to $\langle \mathcal{A}(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})$.

Then, Claim 6 follows from the following sequence of inequalities.

– By the WI property of the left **WIPoK-2** and the definition of $p^{\mathsf{Sim}}_{\mathsf{pref}}$ (Def. 15), it holds that:

$$\Pr\big[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}'_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda).$$

– By the PoK property (per Def. 12) of the left **WIPoK-1** and the above inequality, it holds that:

$$\Pr\big[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}''_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda).$$

– By the WI property of the right **WIPoK-1** and the above inequality, it holds that:

$$\forall i \in [\widetilde{t}],\ \ \Pr\big[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}''_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda).$$

– By the PoK property of the left **WIPoK-1** and the above inequality, it holds that:

$$\forall i \in [\widetilde{t}],\ \ \Pr\big[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda).$$

– By the WI property of the left **WIPoK-2** and the above inequality, it holds that:

$$\forall i \in [\widetilde{t}],\ \ \Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda).$$

This finishes the proof of Claim 6.

□

Machine $\mathcal{K}_i$ $(i \in [\widetilde{t}])$: (Illustrated in Fig. 8a.) For a prefix pref, $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to the $\mathcal{G}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ depicted in Fig. 6, except for the following difference. Machine $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ uses the witness-extended emulator $\mathcal{WE}$ (as per Def. 12) to finish the right **WIPoK-2**, instead of playing the role of the honest receiver.

$\mathcal{K}_i$'s Output: Let w' denote the third output of the witness-extended emulator $\mathcal{WE}$ (see Def. 12), which is supposed to be the witness used by $\mathcal{M}$ in the right **WIPoK-2** (for the statement $(\widetilde{\mathsf{com}}, \widetilde{Y})$ w.r.t. the language $\mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$). Depending on the value of w', we define a value $\mathsf{Val} \in \{0,1\}^{\ell(\lambda)} \cup \{\bot_{\widetilde{Y}}, \bot_{\mathsf{invalid}}\}$ as follows:

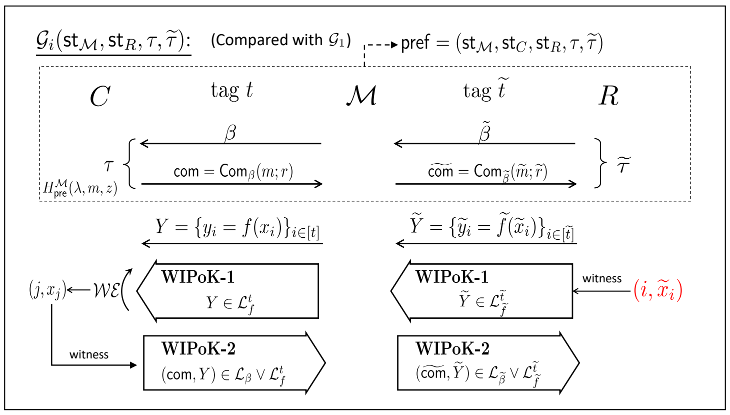


Fig. 6: Machine $\mathcal{G}_i$ (Difference with $\mathcal{G}_1$ is highlighted in red color)

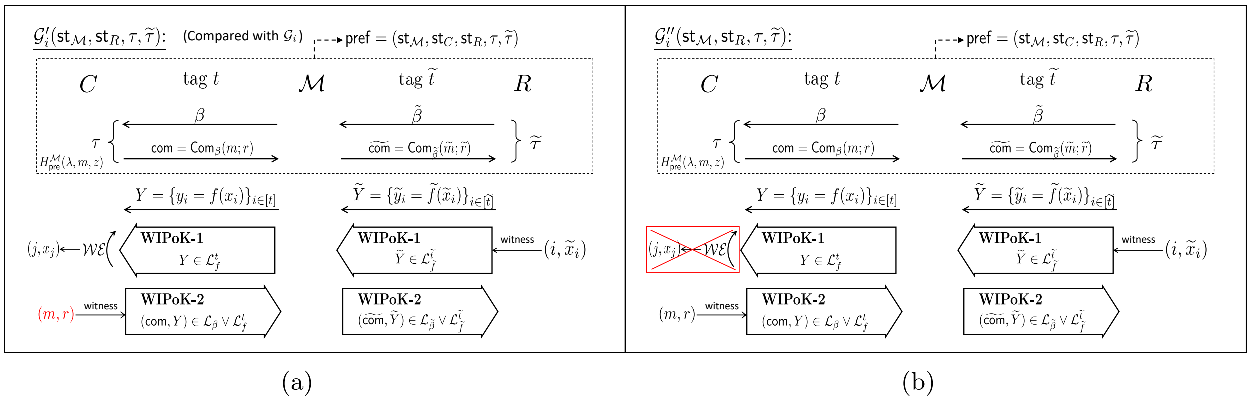


Fig. 7: Machines $\mathcal{G}'_i$ and $\mathcal{G}''_i$ (Difference is highlighted in red color)

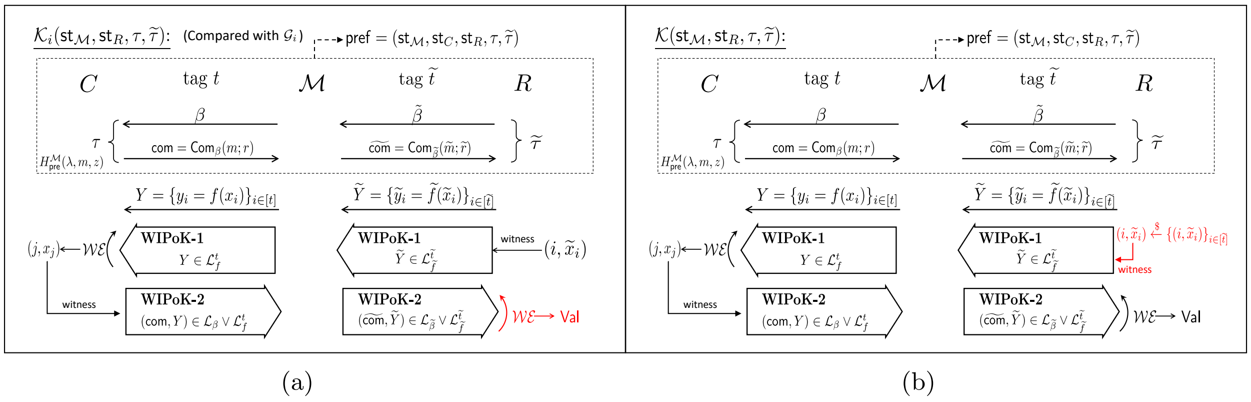


Fig. 8: Machines $\mathcal{K}_i$ and $\mathcal{K}$ (Difference is highlighted in red color)

1. If w' is a valid witness for $(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}_{\widetilde{f}}^{\widetilde{t}}$, then there are tow sub-cases:

(a) w' is a valid witness for $\widetilde{\mathsf{com}} \in \mathcal{L}_{\widetilde{\beta}}$. In this case[22], w' consists of the value $\mathsf{val}(\widetilde{\tau})$, i.e., the value committed in $\widetilde{\tau} = (\widetilde{\beta}, \widetilde{\mathsf{com}})$, *and* the randomness $\widetilde{r}$. We set $\mathsf{Val} \coloneqq \mathsf{val}(\widetilde{\tau})$. Importantly, notice that we do *not* include the randomness $\widetilde{r}$ in Val (as explained in Rmk. 8.).

(b) w' is a valid witness for $\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$: In this case, we set $\mathsf{Val} \coloneqq \bot_{\widetilde{Y}}$.

2. Otherwise, set $\mathsf{Val} \coloneqq \bot_{\mathsf{invalid}}$.

The output of $\mathcal{K}_i$ is defined to be the above Val. Notice that this is in contrast to all previous machines, for which the output is defined to be the man-in-the-middle $\mathcal{M}$'s output and the honest receiver's decision bit. We emphasize that such a Val satisfies the syntax requirement in Property 1 of Lem. 9. In particular, $\mathsf{Val} = \mathsf{val}(\widetilde{\tau})$ *whenever* $\mathsf{Val} \neq \bot$.[23]

Claim 7. $\forall i \in [\widetilde{t}],\ \Pr[\mathsf{Val} \neq \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda)$.

Proof. This claim follows immediately from Claim 6 and the PoK property (as per Def. 12) of the right **WIPoK-2**.

□

Finally, we are ready to define the extractor $\mathcal{K}$ as required by Lem. 9. Intuitively, $\mathcal{K}$ can be conceived as an average-case version of $\{\mathcal{K}_i\}_{i \in [\widetilde{t}]}$:

Extractor $\mathcal{K}$: (Illustrated in Fig. 8b.) For a prefix pref, $\mathcal{K}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ samples uniformly at random an index $i \xleftarrow{\$} [\widetilde{t}]$, executes $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$, and outputs whatever $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ outputs.

Remark 10 (On the Notation of $\mathcal{K}$). It is worth noting that in the above, we write machine $\mathcal{K}$ as $\mathcal{K}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$, while it was written in Lem. 9 as $\mathcal{K}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$. This is only a cosmetic difference.

Next, we show that the extractor $\mathcal{K}$ satisfies the requirements in Lem. 9.

Running Time of $\mathcal{K}$. Observe that for each $i \in [\widetilde{t}]$, $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ differs from the real man-in-the-middle game only in the following places:

- $(i, \widetilde{x}_i)$ is used in the right **WIPoK-1**;
- the witness-extended emulator $\mathcal{WE}$ is used in the right **WIPoK-2** and the left **WIPoK-1**;
- the left **WIPoK-2** is done using the (j, x_j) extracted by $\mathcal{WE}$ from the left **WIPoK-1**.

Since the witness-extended emulator $\mathcal{WE}$ (as per Def. 12) runs in expected PPT, so does $\mathcal{K}_i$. Thus, $\mathcal{K}$ is expected PPT.

Proving Property 1 of Lem. 9. It is straightforward to see that the $\mathcal{K}$ defined above satisfies the syntax requirement in Lem. 9 (see also Rmk. 10). In particular, we have $\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau})$ whenever $\mathsf{Val}_{\mathcal{K}} \neq \bot$, because this is true for each $\mathcal{K}_i$ by definition (see the paragraph for "$\mathcal{K}_i$'s Output" on Page 35).

[22] Technically, we should argue that $\mathcal{K}_i$ is able to detect if this case is happening. This is easy to do as we explicitly give $\widetilde{\tau}$ as an input to $\mathcal{K}_i$—Upon obtaining w', it can just re-execute Naor's commitment by itself to see if w' is consistent with $\widetilde{\tau}$.

[23] Note that here we defined two types of abortion: $\bot_{\widetilde{Y}}$ and $\bot_{\mathsf{invalid}}$, while Property 1 of Lem. 9 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., $\mathsf{Val} \neq \bot \Leftrightarrow (\mathsf{Val} \neq \bot_{\widetilde{Y}} \wedge \mathsf{Val} \neq \bot_{\mathsf{invalid}})$).

Proving Property 2 of Lem. 9. First, recall that Property 2 requires us to show that for any pref in the support of $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$, if $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$, then it holds that

$$\Pr[\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val}_{\mathcal{K}} \leftarrow \mathcal{K}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq \frac{\varepsilon'(\lambda)}{\widetilde{t}}. \tag{26}$$

Also recall that $\mathcal{K}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ is defined to execute the machine $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ with i uniformly sampled from $[\widetilde{t}]$. Therefore, Inequality (26) can be reduced to the following Lem. 10. We will prove Lem. 10 in Sec. 4.6, which will eventually finish the current proof of Lem. 9.

Lemma 10. *Let* $\varepsilon(\lambda) = \frac{1}{\mathsf{poly}(\lambda)}$ *and* $\varepsilon'(\lambda) = \frac{\varepsilon(\lambda)}{t^2}$. *For any* $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$, *if* $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$, *then there exists an* $i \in [\widetilde{t}]$ *such that*

$$\Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq \varepsilon'(\lambda).$$

4.6 Proof of Lem. 10

Notation. We highly recommend reviewing the "$\mathcal{K}_i$'s Output" part on Page 35 (in particular, the meanings of Val, $\bot_{\widetilde{Y}}$, and $\bot_{\mathsf{invalid}}$) before starting to read this subsection. Recall that $\mathcal{K}_i$'s output Val is determined by the w' output by the witness-extended emulator of the right **WIPoK-2**. In this subsection, we will need to refer to this w', though it is not explicitly included as a part of $\mathcal{K}_i$'s output. Particularly, we will make use of the following notation: whenever we write an expression of the form

$$\Pr\big[\text{Some Event } E_{w'} \text{ about } w' : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big],$$

it should be understood as the probability of $E_{w'}$, where w' refers to the w' generated during the random procedure $\mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$, over which the probability is taken.

Using these notations, we can partition the event $\mathsf{Val} = \bot_{\widetilde{Y}}$ as the following *mutually exclusive* events: $w' = (1, \widetilde{x}_1)$, ..., $w' = (\widetilde{t}, \widetilde{x}_{\widetilde{t}})$, where $\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)$ for each $i \in [\widetilde{t}]$. Formally, we express this relation by

$$\Pr\big[\mathsf{Val} = \bot_{\widetilde{Y}} : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] = \sum_{i=1}^{\widetilde{t}} \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \tag{27}$$

With the above notations, we prove Lem. 10 in the following.

Proof for Lem. 10. We assume for contradiction that for some pref satisfying $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$, it holds that

$$\forall i \in [\widetilde{t}], \;\; \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] < \varepsilon'(\lambda). \tag{28}$$

Claim 8. *Under the assumption in Inequality (28), it holds that*

$$\forall i \in [\widetilde{t}], \;\; \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon'(\lambda) - \mathsf{negl}(\lambda).$$

Proof. In this proof, all the probabilities are taken over the random procedure $\mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$.

First, notice that

$$\begin{aligned} \forall i \in [\widetilde{t}], \;\; \Pr[\mathsf{Val} \neq \bot_{\mathsf{invlid}}] &= \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] + \Pr\big[\mathsf{Val} = \bot_{\widetilde{Y}}\big] \\ (\text{by Eq. (27)}) \;\; &= \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] + \Pr\big[w' = (i, \widetilde{x}_i)\big] + \sum_{j \in [\widetilde{t}] \setminus \{i\}} \Pr\big[w' = (j, \widetilde{x}_j)\big]. \end{aligned} \tag{29}$$

Then, the following holds:

$$\forall i \in [\widetilde{t}],\ \ \Pr[w' = (i, \widetilde{x}_i)] = \Pr[\mathsf{Val} \neq \bot_{\mathsf{invalid}}] - \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] - \Bigg(\sum_{j \in [\widetilde{t}] \setminus \{i\}} \Pr[w' = (j, \widetilde{x}_j)]\Bigg) \tag{30}$$

$$\geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda) - \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] - \Bigg(\sum_{j \in [\widetilde{t}] \setminus \{i\}} \Pr[w' = (j, \widetilde{x}_j)]\Bigg) \tag{31}$$

$$\geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda) - \varepsilon'(\lambda) - \Bigg(\sum_{j \in [\widetilde{t}] \setminus \{i\}} \Pr[w' = (j, \widetilde{x}_j)]\Bigg), \tag{32}$$

where Eq. (30) follows from Eq. (29), Inequality (31) follows from Claim 7, and Inequality (32) follows from Inequality (28).

Now, to prove Claim 8, it suffices to show that

$$\forall i \in [\widetilde{t}],\ \forall j \in [\widetilde{t}] \setminus \{i\},\ \ \Pr\big[w' = (j, \widetilde{x}_j) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] = \mathsf{negl}(\lambda). \tag{33}$$

This can be reduced via standard techniques to the one-wayness of the OWF $\widetilde{f}$ in Step 3 of the right execution. In more details, we assume for contradiction that there exist $i^*, j^* \in [\widetilde{t}]$ such that $i^* \neq j^*$ and that the probability $\Pr\big[w' = (j^*, \widetilde{x}_{j^*}) : \mathsf{Val} \leftarrow \mathcal{K}_{i^*}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big]$ is non-negligible, where, by definition, $\widetilde{x}_{j^*}$ is the preimage of $\widetilde{y}_{j^*}$ under the right OWF $\widetilde{f}$. Then, we can build a PPT adversary $\mathcal{A}_{\mathrm{OWF}}$ breaking one-wayness in the following way: $\mathcal{A}_{\mathrm{OWF}}$ obtains the challenge y^* from the external one-wayness challenger; it then runs the machine $\mathcal{K}_{i^*}(\mathsf{pref})$ internally, for which $\mathcal{A}_{\mathrm{OWF}}$ uses y^* in place of $\widetilde{y}_{j^*}$ when executing Step 3 in the right. Note that the internal execution of $\mathcal{K}_{i^*}(\mathsf{pref})$ is identically to the real execution of $\mathcal{K}_{i^*}$, thus the extracted $w' = (j^*, \widetilde{x}_{j^*})$ must satisfy $\widetilde{f}(\widetilde{x}_{j^*}) = \widetilde{y}^*_j\ (= y^*)$ with non-negligible probability, breaking one-wayness.

This finishes the proof of Claim 8.

□

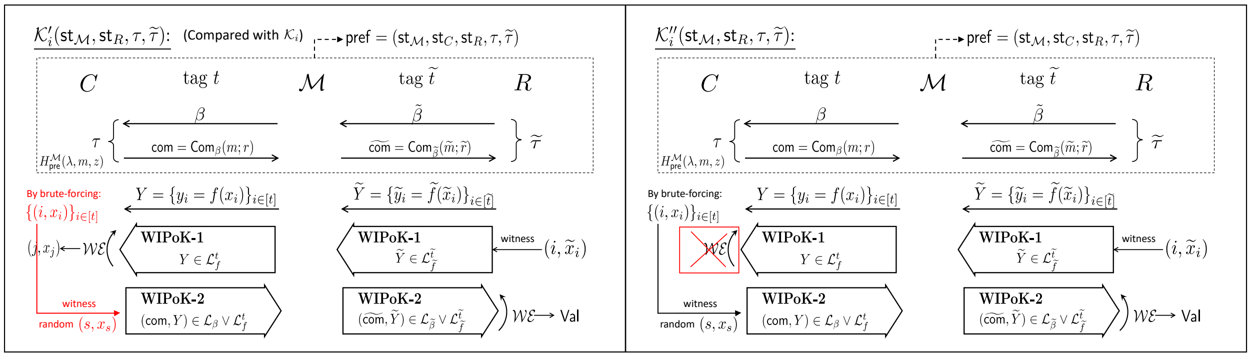
Machine $\mathcal{K}'_i$ ($i \in [\widetilde{t}]$): (Illustrated in Fig. 9a.) For a prefix pref, $\mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ proceeds as follows:

1. It first finishes Step 3 of the man-in-the-middle execution in the same way as $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$. In particular, it will see in the left execution the values $Y = (y_1, \ldots, y_t)$ sent from $\mathcal{M}$;
2. It then recovers $(x_1, \ldots, x_t)$ by brute-force: Namely, for each $i \in [t]$, it inverts the OWF f to find x_i s.t. $f(x_i) = y_i$. It is possible that there exist some "bad" y_i's that are not in the range of f. For such bad i's, it sets $x_i = \bot$. If all the x_i's are bad, $\mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ halts and outputs Fail;
3. If this step is reached, we know that $(x_1, \ldots, x_t)$ cannot be all-$\bot$. $\mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ then picks an (s, x_s) uniformly at random from the good (i.e. non-$\bot$) x_i's.
4. Then, $\mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ continues to finish the execution in the same way as $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$, except that it uses (s, x_s) as the witness when executing the left **WIPoK-2**.

It is worth noting that the (j, x_j) extracted by the witness-extended emulator for the left **WIPoK-1** (inherited from $\mathcal{K}_i(\mathsf{pref})$) is not used any more in $\mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$.

Obviously, if the (s, x_s) picked by $\mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ is equal to the (j, x_j) extracted in $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ from its left **WIPoK-1**, then $\mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ and $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ are identical. Notice that this step relies on the injectivity of f (see Rmk. 11). Since $\mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ samples (s, x_s) uniformly from all the good (i, x_i)'s, it must hold with probability at least $1/t$ that $(s, x_s) = (j, x_j)$. Therefore, the following must hold:

$$\forall i \in [\widetilde{t}],\ \ \Pr[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq \frac{1}{t} \cdot \Pr[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})]. \tag{34}$$



$\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
(Compared with $\mathcal{K}_i$)
$\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\widetilde{\beta}$
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
τ
$\widetilde{\tau}$
$H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$
By brute-forcing:
$\{(i, x_i)\}_{i \in [t]}$
$Y = \{y_i = f(x_i)\}_{i \in [t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i \in [\widetilde{t}]}$
$(j, x_j) \leftarrow \mathcal{WE}$
WIPoK-1
$Y \in \mathcal{L}^t_f$
$\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
witness
$(i, \widetilde{x}_i)$
WIPoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
random (s, x_s)
$\mathcal{WE} \longrightarrow \mathsf{Val}$
$\mathcal{K}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:

(a) (b)

Fig. 9: Machines $\mathcal{K}'_i$ and $\mathcal{K}''_i$ (Difference is highlighted in red color)

Remark 11 (On Injectivity of the OWF). We emphasize that throughout the proof of non-malleability, this is *the only* place where we rely on the injectivity of the OWF. In particular, we rely on the injectivity of the f in the left session to ensure that there is a unique preimage for each $\{y_i\}_{i\in[t]}$. Thus, if both the x_s with $s = j$ (sampled by $\mathcal{K}'_i$) and the extracted x_j (in $\mathcal{K}_i$) are a valid preimage for the same y_j, then the injectivity of f implies that $x_s = x_j$. We will show how to remove injectivity in Sec. 4.8.

Machine $\mathcal{K}''_i$ ($i \in [\widetilde{t}]$): (Illustrated in Fig. 9b.) For a prefix pref, $\mathcal{K}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it plays as the honest committer in the left **WIPoK-1**, instead of running the witness-extended emulator. Recall that starting from $\mathcal{K}'_i$, the witness (j, x_j) extracted by the witness-extended emulator from the left **WIPoK-1** is not used any more; Thus, machine $\mathcal{K}''_i$ does not need to perform this witness-extended emulation.

By the PoK property of the left **WIPoK-1**, it holds that

$$\forall i \in [\widetilde{t}],~ \Big|\Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] - \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big]\Big| \le \mathsf{negl}(\lambda). \tag{35}$$

Next, by the (non-uniform, see Rmk. 12) WI property of the right **WIPoK-1**, it holds that

$$\forall i \in [\widetilde{t}],~ \Big|\Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] - \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big]\Big| \le \mathsf{negl}(\lambda). \tag{36}$$

Remark 12 (Power of Non-Uniform Reductions). Note that we can rely on the PoK and WI properties, although $\mathcal{K}'_i$ and $\mathcal{K}''_i$ perform brute-force to recover $(x_1, ..., x_t)$. This is because the brute-forcing step is done before **WIPoK-1** or **WIPoK-2** starts; Thus, $(x_1, ..., x_t)$ can be treated as a non-uniform advice in the reductions. This non-uniform type of argument will be used again in this section later.

Then, we have the following claim:

Claim 9. $\forall i \in [\widetilde{t}],~ \Pr[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] \ge \frac{1}{t} \cdot \big(p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon'(\lambda)\big) - \mathsf{negl}(\lambda)$.

Proof. This claim follows from Claim 8 and Inequalities (34) to (36). Formally, (in the following, we omit the input $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ to $\mathcal{K}_i$, $\mathcal{K}'_i$, and $\mathcal{K}''_i$)

$$\begin{aligned} \forall i \in [\widetilde{t}],\ \ \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_1\big] &\geq \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_i\big] - \mathsf{negl}(\lambda) && (37)\\ &\geq \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}'_i\big] - \mathsf{negl}(\lambda) && (38)\\ &\geq \frac{1}{t} \cdot \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i\big] - \mathsf{negl}(\lambda) && (39)\\ &\geq \frac{1}{t} \cdot \big(p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon'(\lambda)\big) - \mathsf{negl}(\lambda), && (40) \end{aligned}$$

where Inequality (37) follows from Inequality (36), Inequality (38) follows from Inequality (35), Inequality (39) follows from Inequality (34), and Inequality (40) follows from Claim 8.

□

Now, we make the last claim which, together with Claim 9, leads to the desired contradiction.

Claim 10. $\Pr[\mathsf{Val} \neq \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \leq p^{\mathsf{Sim}}_{\mathsf{pref}} + \mathsf{negl}(\lambda)$.

Deriving the Contradiction. Before proving Claim 10, we first show why Claims 9 and 10 are contradictory (all the probabilities below are taken over $\mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$):

$$\begin{aligned} \Pr[\mathsf{Val} \neq \bot_{\mathsf{invalid}}] &= \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] + \Pr\big[\mathsf{Val} = \bot_{\widetilde{Y}}\big] \\ &= \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] + \sum_{i=1}^{\widetilde{t}} \Pr\big[w' = (i, \widetilde{x}_i)\big] && (41)\\ &\geq \sum_{i=1}^{\widetilde{t}} \Pr\big[w' = (i, \widetilde{x}_i)\big] \\ &\geq \widetilde{t} \cdot \frac{1}{t} \cdot \big(p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon'(\lambda)\big) - \mathsf{negl}(\lambda) && (42)\\ &\geq (1 + \frac{1}{t}) \cdot (p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon'(\lambda)) - \mathsf{negl}(\lambda) && (43)\\ &= p^{\mathsf{Sim}}_{\mathsf{pref}} + \left(\frac{p^{\mathsf{Sim}}_{\mathsf{pref}}}{t} - \varepsilon'(\lambda) - \frac{\varepsilon'(\lambda)}{t}\right) - \mathsf{negl}(\lambda) \\ &\geq p^{\mathsf{Sim}}_{\mathsf{pref}} + \frac{10t^2 - t - 1}{10t^3} \cdot \varepsilon(\lambda) - \mathsf{negl}(\lambda), && (44) \end{aligned}$$

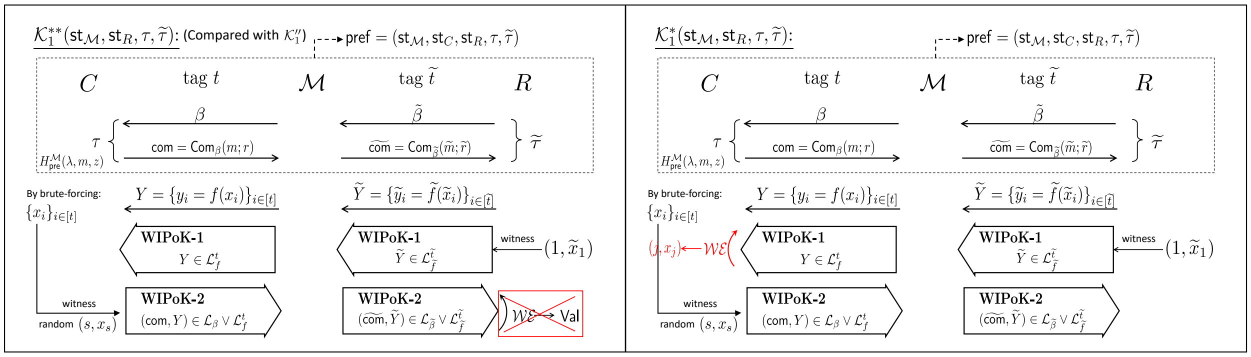
where Eq. (41) follows from Eq. (27), Inequality (42) follows from Claim 9, Inequality (43) follows from the assumption that $\widetilde{t} \geq t+1$, and Inequality (44) follows from the assumption that $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$ and our parameter setting $\varepsilon'(\lambda) = \frac{\varepsilon(\lambda)}{10t^2}$.

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 (44) can be written as:

$$\Pr\big[\mathsf{Val} \neq \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{pref})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} + \frac{1}{\mathsf{poly}(\lambda)} - \mathsf{negl}(\lambda),$$

which contradicts Claim 10.

This eventually finishes the proof of Lem. 10 (modulo the proof of Claim 10, which we show in Sec. 4.7).



$\mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$: (Compared with $\mathcal{K}_1''$)
pref = $(\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\widetilde{\beta}$
τ
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
$\widetilde{\tau}$
$H^\mathcal{M}_\mathsf{pre}(\lambda, m, z)$
By brute-forcing:
$\{x_i\}_{i\in[t]}$
$Y = \{y_i = f(x_i)\}_{i\in[t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i\in[\widetilde{t}]}$
WIPoK-1
$Y \in \mathcal{L}_f^t$
$\widetilde{Y} \in \mathcal{L}_{\widetilde{f}}^{\widetilde{t}}$
witness
$(1, \widetilde{x}_1)$
WIPoK-2
random (s, x_s)
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}_f^t$
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}_{\widetilde{f}}^{\widetilde{t}}$
$\mathcal{WE}$ → Val
$\mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
$(j, x_j) \leftarrow \mathcal{WE}$
(a)
(b)

Fig. 10: Machines $\mathcal{K}_1^{**}$ and $\mathcal{K}_1^{*}$ (Difference is highlighted in red color)

4.7 Proof of Claim 10

We first define two extra machines $\mathcal{K}_1^{**}$ and $\mathcal{K}_1^{*}$.

Machine $\mathcal{K}_1^{**}$: (Illustrated in Fig. 10a.) For a prefix pref, $\mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically as $\mathcal{K}_1''(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that $\mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ finishes the right **WIPoK-2** using the honest receiver's algorithm, instead of using the witness-extended emulator.

$\mathcal{K}_1^{**}$'s Output. We define the output of $\mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ to be the honest receiver's decision $b \in \{\top, \bot\}$. This is in contrast to previous hybrids $\mathcal{K}_i'$, $\mathcal{K}_i''$, and $\mathcal{K}_i$, whose output is defined to be the value Val that depends on the value w' extracted by the $\mathcal{WE}$ for the right **WIPoK-2**.

By the (non-uniform) PoK property (as per Def. 12) of the right **WIPoK-2**, it holds that

$$\Big|\Pr\big[\mathsf{Val} \neq \bot_\mathsf{invalid} : \mathsf{Val} \leftarrow \mathcal{K}_1''(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] - \Pr[b = \top : b \leftarrow \mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]\Big| \leq \mathsf{negl}(\lambda). \quad (45)$$

Machine $\mathcal{K}_1^{*}$: (Illustrated in Fig. 10b.) For a prefix pref, $\mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically as $\mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except the following difference: $\mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ uses the witness-extended emulator to extract a witness (j, x_j) from the left **WIPoK-1**, and if x_j is not a valid preimage for y_j, $\mathcal{K}_1^{*}$ aborts.

By the (non-uniform) PoK property of the left **WIPoK-1**, it holds that

$$\Big|\Pr[b = \top : b \leftarrow \mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] - \Pr[b = \top : b \leftarrow \mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]\Big| \leq \mathsf{negl}(\lambda). \quad (46)$$

Compare $\mathcal{K}_1^{*}$ with $\mathcal{G}_1$. Now, let us compare $\mathcal{K}_1^{*}$ with the $\mathcal{G}_1$ depicted in Fig. 6. They only differ in the witness used in the left **WIPoK-2** (and that $\mathcal{G}_1$ does not need to perform brute-forcing for Y, as it does not use those preimages). Therefore, by the (non-uniform) WI property of the left **WIPoK-2**, it holds that

$$\Big|\Pr[b = \top : b \leftarrow \mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] - \Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]\Big| \leq \mathsf{negl}(\lambda).$$

Also, recall (from Def. 15) that $\Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]$ is exactly the definition of $p^\mathsf{Sim}_\mathsf{pref}$. Thus, the above implies:

$$\Big|\Pr[b = \top : b \leftarrow \mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] - p^\mathsf{Sim}_\mathsf{pref}\Big| \leq \mathsf{negl}(\lambda). \quad (47)$$

Therefore, the following holds:

$$\begin{aligned}\Pr\left[\mathsf{Val} \neq \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}_1''(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\right] &\leq \Pr[b = \top : b \leftarrow \mathcal{K}_1^{**}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] + \mathsf{negl}(\lambda) \quad (48)\\ &\leq \Pr[b = \top : b \leftarrow \mathcal{K}_1^{*}(\mathsf{pref})] + \mathsf{negl}(\lambda) \quad (49)\\ &\leq p^{\mathsf{Sim}}_{\mathsf{pref}} + \mathsf{negl}(\lambda), \quad (50)\end{aligned}$$

where Inequality (48) follows from Inequality (45), Inequality (49) follows from Inequality (46), and Inequality (50) follows from Inequality (47).

This finishes the proof of Claim 10.

4.8 Replacing the Injective OWF with Any OWF

In this part, we show how to replace the injective OWF with any OWF in Prot. 1.

As mentioned in Rmk. 11, the injectivity is used only when we switch from $\mathcal{K}_i$ to $\mathcal{K}'_i$ (on Page 39). Let us first recall why injectivity is important there: $\mathcal{K}'_i$ learns the preimages of each $\{y_i\}_{i\in[t]}$ via brute-forcing, while $\mathcal{K}_i$ learns a preimage x_j of y_j *for some* $j \in [t]$ by extraction (from the left **WIPoK-1**). Our goal there is to show that if $\mathcal{K}'_i$ picks a random preimage, then it will hit the (j, x_j) extracted by $\mathcal{K}_i$, except for a multiplicative $1/t$ loss on the probability (i.e., Inequality (34)). If the OWF f is not injective, there is no guarantee that the (j, x_j) extracted by $\mathcal{K}_i$ is really contained in the set of all preimages learned by $\mathcal{K}'_i$ via brute-forcing—Even if $\mathcal{K}'_i$ guessed the index j correctly, it could have brute-forced a $x'_j \neq x_j$ (though it holds that $y_j = f(x_j) = f(x'_j)$). In this case, Inequality (34) may not hold anymore.

From the above discussion, it is clear that if we want to remove the requirement of injectivity, we only need to find an alternative approach to ensure that the preimage extracted by $\mathcal{K}_i$ will fall in the set of the preimages learned by $\mathcal{K}'_i$ via brute-forcing. It turns out that this can be achieved by the following simple modification to Steps 3 and 4 of Prot. 1:

– In Step 3, ask R to *additionally* commit (using a statistically-binding scheme) to the preimages $\{x_i\}_{i\in[t]}$ that it used to generate $\{y_i\}_{i\in[t]}$. Formally, we ask R to send the following messages in Step 3:

$$Y = \{y_i = f(x_i)\}_{i\in[t]}, \text{ and } \{\mathsf{com}_i = \mathsf{Com}_{\beta'}(x_i; r_i)\}_{i\in[t]},$$

where $\mathsf{Com}_{\beta'}$ is Naor's commitment with β' being the first Naor's message from C (which could be sent in parallel with other messages in Step 2).

– In Step 4, ask R to additionally prove that the Y and $\{\mathsf{com}_i\}_{i\in[t]}$ are "consistent". Formally, instead of proving $Y \in \mathcal{L}^t_f$ (defined in Language (10)), we ask R to prove $(Y, \{\mathsf{com}_i\}_{i\in[t]}) \in \mathcal{L}^t_{f,\beta'}$ using the WIPoK, where

$$\mathcal{L}^t_{f,\beta'} \coloneqq \{(y_1, \ldots, y_t), (\mathsf{com}_1, \ldots, \mathsf{com}_t) \mid \exists (i, x_i, r_i)\ s.t.\ i \in [t] \wedge y_i = f(x_i) \wedge \mathsf{com}_i = \mathsf{Com}_{\beta'}(x_i; r_i)\}. \quad (51)$$

It is worth noting that we do *not* need to modify Step 5. In particular, C does *not* need to prove that it knows one of the (x_i, r_i)'s used by R to generate $\{\mathsf{com}_i\}_{i\in[t]}$; C still only needs to prove $(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$.

Now, instead of performing brute-force over $\{y_i\}_{i\in[t]}$, $\mathcal{K}'_i$ does that over $\{\mathsf{com}_i\}_{i\in[t]}$ to learn the committed values, and then picks a random x_s satisfying $y_s = f(x_s)$ to use in the left **WIPoK-2**.[24] It is easy to see that the statistical-binding property of Com ensures that Inequality (34) still holds for an overwhelming fraction of β'.

[24] Note that $\mathcal{K}'_i$ (resp. $\mathcal{K}_i$) will also learn the randomness used to generate the commitment via brute-forcing (resp. via extraction from **WIPoK-1**). They simply ignore the randomness—indeed, both machines only need to know a preimage (j, x_j) to finish **WIPoK-2**; the corresponding randomness r_i is irrelevant.

Remark 13 (On One-Wayness). Technically, we also need to argue that the above modification does not enable the man-in-the-middle $\mathcal{M}$ to invert the one-way function $\widetilde{f}$ on $\widetilde{Y} = \{\widetilde{y}_i\}_{i\in[\widetilde{t}]}$. (Recall that the one-wayness of $\widetilde{f}$ is used (only) in the proof of Claim 8.) This is straightforward—Any OWF remains one-way even if a commitment to its preimage is given.

4.9 Summarizing the Proof Strategy for Thm. 2

Since the proof of Thm. 2 is complicate, we summarize the logic flow in this subsection. We hope that it will help the reader obtain a better understanding of the whole picture, without being disturbed by details of secondary importance. But one could safely skip this subsection as it does not contain new information. In particular, it will not affect the understanding of succeeding sections.

At a high level, the proof of Thm. 2 can be divided into the following five steps:

1. **(Proof by Contradiction.)** In the proof of Thm. 2, we first denoted the MIM execution by $H^{\mathcal{M}}(\lambda, m, z)$ (on Page 26). We then assumed for contradiction that a (potentially non-uniform) PPT adversary can distinguish the outputs of $H^{\mathcal{M}}(\lambda, m_0, z)$ and $H^{\mathcal{M}}(\lambda, m_1, z)$ with advantage $3 \cdot \delta(\lambda)$, with $\delta(\lambda)$ being some polynomial (i.e., Inequality (15)). Next, we required the existence of a machine $G^{\mathcal{M}}$ (i.e., Lem. 6) such that

   (a) the outputs of $G^{\mathcal{M}}(\lambda, m_0, z)$ and $G^{\mathcal{M}}(\lambda, m_1, z)$ are computationally indistinguishable, and

   (b) $G^{\mathcal{M}}(\lambda, m_b, z)$ and $H^{\mathcal{M}}(\lambda, m_b, z)$ is (computationally) $\delta(\lambda)$-close for both $b = 0$ and $b = 1$.

   The existence of such a $G^{\mathcal{M}}$ finished the proof of Thm. 2, because it contradicts the assumption that there exists some PPT distinguisher that tells the difference between $H^{\mathcal{M}}(\lambda, m_0, z)$ and $H^{\mathcal{M}}(\lambda, m_1, z)$ with advantage $3 \cdot \delta(\lambda)$. Thus, the remaining steps for the proof of Thm. 2 were then devoted to constructing such a $G^{\mathcal{M}}$.

2. **(Constructing $G^{\mathcal{M}}$.)** To build the $G^{\mathcal{M}}$ required in Step 1, we first broke $H^{\mathcal{M}}$ into two stages—the prefix part $H^{\mathcal{M}}_{\mathsf{pre}}$ and the remainder $\langle \mathcal{A}, \mathcal{B} \rangle$ (i.e., Algo. 4.1). Here, we defined an important variable called "prefix":

   – An execution of $H^{\mathcal{M}}_{\mathsf{pre}}$ fixes the prefix pref, which contains the internal state $\mathsf{st}_C, \mathsf{st}_{\mathcal{M}}, \mathsf{st}_R$ of the corresponding parties at the end of **Step-2** in a MIM execution. Also, pref fixes the right Naor's commitment $\widetilde{\tau}$. Also note that the committed value in the right execution was defined to be $\mathsf{val}_b(\widetilde{\tau})$ (i.e., Eq. (12)).

   Then, we constructed a simulation-extractor $\mathcal{SE}^{\mathcal{M}}$ (in Lem. 7) such that, when continuing from any prefix pref (Importantly, $\mathcal{SE}$ cannot use the intermediate state st_C for the committer; Otherwise, we cannot show Property 1a by reducing it to the hiding property of Naor's commitment), $\mathcal{SE}^{\mathcal{M}}$ will ε-simulate the remainder execution $\langle \mathcal{A}, \mathcal{B} \rangle$, while also extracting the committed value $\mathsf{val}_b(\widetilde{\tau})$ in the right execution.
   With this $\mathcal{SE}^{\mathcal{M}}$, it is easy to construct $G^{\mathcal{M}}$—$G^{\mathcal{M}}$ simply runs $H^{\mathcal{M}}_{\mathsf{pre}}$ to obtain the prefix, and then runs $\mathcal{SE}$ to simulate the remainder execution and also extract $\mathsf{val}_b(\widetilde{\tau})$. Thus, the remaining steps for the proof of Thm. 2 were then devoted to constructing such a $\mathcal{SE}^{\mathcal{M}}$.

3. **(Constructing $\mathcal{SE}^{\mathcal{M}}$.)** To construct $\mathcal{SE}^{\mathcal{M}}$, we first built two machines:

   (a) a simulator $\mathcal{G}_1$ (in Lem. 8) that can simulate the execution $\langle \mathcal{A}, \mathcal{B} \rangle$ without knowing C's internal states st_C. This is to address the point that $\mathcal{SE}^{\mathcal{M}}$ is not allowed to use st_C. Here, we defined an important variable $p^{\mathsf{Sim}}_{\mathsf{pref}}$, which denotes the probability that the honest R is convinced in the execution simulated by $\mathcal{G}_1$.

(b) an extractor $\mathcal{K}$ (in Lem. 9) such that for any prefix pref and noticeable $\varepsilon(\lambda)$, if $p^{\mathsf{Sim}}_{\mathsf{pref}} \ge \varepsilon(\lambda)$, then $\mathcal{K}$ can extract the value committed in $\widetilde{\tau}$ (contained in pref) with probability polynomially related to $\varepsilon(\lambda)$ (more accurately, the probability is $\frac{\varepsilon'(\lambda)}{\widetilde{t}}$ where $\varepsilon'(\lambda) := \frac{\varepsilon(\lambda)}{10t^2}$).

With $\mathcal{G}_1$ and $\mathcal{K}$, $\mathcal{SE}^{\mathcal{M}}$ can be constructed as follows: on input a pref, it first runs $\mathcal{G}_1$ to simulate the main-tread execution of $\langle \mathcal{A}, \mathcal{B} \rangle$; If the honest receiver is not convinced in the main-thread, $\mathcal{SE}^{\mathcal{M}}$ simply sets the committed value in the right execution as $\bot$; Otherwise, $\mathcal{SE}^{\mathcal{M}}$ will extract the value committed in $\widetilde{\tau}$ (denoted as $\mathsf{val}(\widetilde{\tau})$) by repeating $\mathcal{K}$ for polynomially many times (more accurately, $\frac{\widetilde{t}}{\varepsilon'(\lambda)} \cdot \lambda$ times). As shown toward the end of Sec. 4.4, such a $\mathcal{SE}^{\mathcal{M}}$ satisfied the requirements described in Step 2. Thus, the remaining steps for the proof of Thm. 2 were then devoted to constructing the extractor $\mathcal{K}$. (The construction and security of $\mathcal{G}_1$ is relatively straightforward, so we do not repeat it in this summary.)

4. **(Constructing $\mathcal{K}$.)** To construct $\mathcal{K}$, we first defined a sequence of machines $\{\mathcal{K}_i\}_{i \in [\widetilde{t}]}$ as shown in Fig. 8a. We then argued that there must exist an $i \in [\widetilde{t}]$ (called "the good i") such that the Val output by $\mathcal{K}_i$ must be $\mathsf{val}(\tau)$ with probability $\ge \varepsilon'(\lambda)$. Then, machine $\mathcal{K}$ was defined to pick a random $i \xleftarrow{\$} [\widetilde{t}]$ and run machine $\mathcal{K}_i$. Since such a $\mathcal{K}$ will hit the good i with probability $1/\widetilde{t}$, it will successfully extract $\mathsf{val}(\widetilde{\tau})$ with probability $\frac{\varepsilon'(\lambda)}{\widetilde{t}}$, satisfying the requirement in Step 3. Now, to finish the proof of Thm. 2, the only thing left is to prove that $\{\mathcal{K}_i\}_{i \in [\widetilde{t}]}$ satisfies the above requirement.

5. **(Existence of Good $\mathcal{K}_i$.)** Informally, our current goal is to show the following (see Lem. 10 for the precise statement):

$$\exists i \in [\widetilde{t}] \;\; s.t. \;\; \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i] \ge \varepsilon'(\lambda).$$

We first assumed for contradiction that (i.e., Inequality (28)):

$$\forall i \in [\widetilde{t}], \;\; \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i] \le \varepsilon'(\lambda). \tag{52}$$

We then derived the contradiction in the following way:

(a) We first showed that the Val output by $\mathcal{K}_i$ must be a valid witness with probability $\ge p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda)$ (i.e., Claim 7). Next, observe that in $\mathcal{K}_i$, Val cannot be $(j, \widetilde{x}_j)$ for some $j \ne i$; Otherwise, it breaks the one-wayness of the right OWF $\widetilde{f}$. Thus, when Val is a valid witness in $\mathcal{K}_i$, it can only take the values $\mathsf{val}(\widetilde{\tau})$ or $(i, \widetilde{x}_i)$. This together with our assumption in Inequality (52) implies that

$$\forall i \in [\widetilde{t}], \;\; \Pr[\mathsf{Val} = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i] \ge p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon'(\lambda) - \mathsf{negl}(\lambda). \tag{53}$$

(This is exactly Claim 8.)

(b) Next, we defined machine $\mathcal{K}''_i$ (see Fig. 9b), which differs from $\mathcal{K}_i$ in the following way: $\mathcal{K}''_i$ does not rewind the left **WIPoK-1**. Instead, it extracts the preimages of the right $\{y_i\}_{i \in [t]}$ by brute force, and then picks a random preimage (s, x_s) to use in the left **WIPoK-2**. Since the (s, x_s) guessed by $\mathcal{K}''_i$ will hit the one used by $\mathcal{K}_i$, the following holds:

$$\forall i \in [\widetilde{t}], \;\; \Pr\big[\mathsf{Val} = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_i\big] \ge \frac{1}{t} \cdot \Pr[\mathsf{Val} = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i]. \tag{54}$$

(This is essentially Inequality (34)—Note that in Sec. 4.6, we also defined a machine $\mathcal{K}'_i$. But it is only a tool that helps us eventually connect $\mathcal{K}''_i$ and $\mathcal{K}_i$. So we ignore $\mathcal{K}'_i$ in this relatively high-level summary.)

(c) Since in $\mathcal{K}''_i$ we do not rewind the left **WIPoK-1** anymore, we can now rely on the WI property of the right **WIPoK-1** to connect all the $\{\mathcal{K}''_i\}_{i\in[\widetilde{t}]}$ to $\mathcal{K}''_1$. Namely, we have

$$\forall i \in [\widetilde{t}], \ \left|\Pr\left[\mathsf{Val} = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_i\right] - \Pr\left[\mathsf{Val} = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_1\right]\right| \le \mathsf{negl}(\lambda). \tag{55}$$

(This is exactly Inequality (36).) We want to remark that in this step, we took advantage of the synchronous schedule: in the synchronous setting, it is guaranteed that the brute-forcing for $\{y_i\}_{i\in[n]}$ happens *before* the right **WIPoK-1**; Otherwise, we will not be able to reduce Inequality (55) to the WI property of the right **WIPoK-1**.

Now, by Inequalities (53) to (55), the following holds (this is exactly Claim 9)

$$\forall i \in [\widetilde{t}], \ \Pr\left[\mathsf{Val} = (i, \widetilde{x}_i) \text{ in } \mathcal{K}''_1\right] \ge \frac{1}{t} \cdot \left(p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon'(\lambda)\right) - \mathsf{negl}(\lambda). \tag{56}$$

This implies

$$\Pr\left[\mathsf{Val} = (1, \widetilde{x}_1) \vee \ldots \vee \mathsf{Val} = (\widetilde{t}, \widetilde{x}_{\widetilde{t}}) : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{pref})\right] \ge \widetilde{t} \cdot \frac{1}{t} \cdot \left(p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon'(\lambda)\right) + \mathsf{negl}(\lambda)$$

$$(\text{By our parameter setting. See Inequality (44)}) \ge p^{\mathsf{Sim}}_{\mathsf{pref}} + \frac{10t^2 - t - 1}{10t^3} \cdot \varepsilon(\lambda) - \mathsf{negl}(\lambda)$$

On the other hand, Claim 10 implies that (we do not repeat the proof of Claim 10 here as it is relatively straightforward)

$$\Pr\left[\mathsf{Val} = (1, \widetilde{x}_1) \vee \ldots \vee \mathsf{Val} = (\widetilde{t}, \widetilde{x}_{\widetilde{t}}) : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{pref})\right] \le p_{\mathsf{pref}} + \mathsf{negl}(\lambda)$$

This gives us the desired contradiction because $\frac{10t^2-t-1}{10t^3} \cdot \varepsilon(\lambda)$ is an (inverse) polynomial on λ.

5 Small-Tag, Synchronous, Classical Setting

In this section, we show how to make $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ (i.e., Prot. 1) secure without the "one-sided" restriction. We emphasize that this section still focuses on the small-tag, synchronous setting against classical (i.e., non-uniform PPT) adversaries.

5.1 High-Level Idea

Our proof for the non-malleability of $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg}}$ (shown in Prot. 1) 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 man-in-the-middle $\mathcal{M}$ 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 technique as in Sec. 4.

To do that, first observe that the only place where Prot. 1 makes use of the tag t is Steps 3 and 4. Let us call these two messages a "**Slot**". Our modification is to execute this **Slot** twice sequentially, where

– for the first execution (referred to as **Slot-A**), we ask R to use tag t as in the original construction;

– for the second execution (referred to as **Slot-B**), we ask R to use $n - t$ as the tag.

We also modify the language instance proven by **WIPoK-2** in the expected way—Now C needs to prove that it "knows" the (m, r) tuple, or the preimage of one of the y_i's sent in **Slot-A**, *or the preimage of one of the y_i's sent in* **Slot-B**.

By the above design, it is easy to see that one of the following case must happen for any man-in-the-middle execution (note that we are still in the synchronous setting):

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 argument in Sec. 4 to **Slot-A**. We sometimes refer to this case by saying that **Slot-A** *is the good slot.*
3. $t > \widetilde{t}$: In this case, it holds that $(n - t) < (n - \widetilde{t})$. 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 argument in Sec. 4 to **Slot-B**. We sometimes refer to this case by saying that **Slot-B** *is the good slot.*

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

5.2 Construction of $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$

We refer to the construction described in Sec. 5.1 as $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$, and formally present it in Prot. 2. This protocol makes use of the same building blocks as for Prot. 1, namely:

- An OWF f, (Note that we do not require injectivity any more, because Prot. 2 makes use of the technique discussed in Sec. 4.8.)
- Naor's commitment Com;
- A witness-indistinguishable proof of knowledge WIPoK (as per Def. 13).

We remark that we further optimize the "two-slot" approach from Sec. 5.1 by sending the first message of **Slot-A** and the first message of **Slot-B** together.

Protocol 2: Small-Tag Synchronous NMCom $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$

The tag space is defined to be $[n]$ where n is a polynomial on λ. Let $t \in [n]$ be the tag for the following interaction. Let m be the message to be committed to.

Commit Stage:

1. Receiver R samples and sends the first message β for Naor's commitment.
2. Committer C commits to m using the second message of Naor's commitment. C also sends a β' that will be used as the first Naor's message for R to generate a commitment in next step. Formally, C samples r and β' and sends the tuple $\big(\mathsf{com} = \mathsf{Com}_\beta(m; r),\ \beta'\big)$.

   Comment: The β' is to address the injectivity issue as discussed in Sec. 4.8.
3. R performs the following computation:
   (a) R computes $\{y^A_i = f(x^A_i)\}_{i\in[t]}$ with $x^A_i \xleftarrow{\$} \{0,1\}^\lambda$ for each $i \in [t]$, and sets $Y^A := (y^A_1, \ldots, y^A_t)$. R also computes $\{\mathsf{com}^A_i = \mathsf{Com}_{\beta'}(x^A_i; r^A_i)\}_{i\in[t]}$, where r^A_i is a random string for each $i \in [t]$.

   (b) R computes $\{y^B_i = f(x^B_i)\}_{i\in[n-t]}$ with $x^B_i \xleftarrow{\$} \{0,1\}^\lambda$ for each $i \in [n - t]$, and sets $Y^B := (y^B_1, \ldots, y^B_{n-t})$. R also computes $\{\mathsf{com}^B_i = \mathsf{Com}_{\beta'}(x^B_i; r^B_i)\}_{i\in[n-t]}$, where r^B_i is a random string for each $i \in [n - t]$.

   R sends the tuple $\big(Y^A,\ \{\mathsf{com}^A_i = \mathsf{Com}_{\beta'}(x^A_i; r^A_i)\}_{i\in[t]},\ Y^B,\ \{\mathsf{com}^B_i = \mathsf{Com}_{\beta'}(x^B_i; r^B_i)\}_{i\in[n-t]}\big)$.

Comment: Compared with Step 3 of Prot. 1, the current Step 3 has two differences: (1) R additionally commits to the preimages of y_i's. This is to address the injectivity issue as discussed in Sec. 4.8. (2) R sends *two copies* of the $(Y, \{\mathsf{com}_i\}_i)$ tuple, indexed by superscripts A and B respectively. This is corresponding to the "two-slot" approach discussed in Sec. 5.1 (with the first message of **Slot-B** being sent in parallel with the first message of **Slot-A**).

4. **(WIPoK-1-A.)** R and C execute an instance of WIPoK where R proves to C that he "knows" a pre-image of some y_i^A contained in Y^A, and y_i^A are consistent with com_i^A (as defined in Step 3a). Formally, R proves that $(Y^A, \{\mathsf{com}_i^A\}_{i\in[t]}) \in \mathcal{L}^t_{f,\beta'}$, where

$$\mathcal{L}^t_{f,\beta'} := \left\{ (y_1^A, \ldots, y_t^A), (\mathsf{com}_1^A, \ldots, \mathsf{com}_t^A) \;\middle|\; \exists (i, x_i^A, r_i^A) \text{ s.t. } \begin{array}{l} i \in [t] \wedge \\ y_i^A = f(x_i^A) \wedge \\ \mathsf{com}_i^A = \mathsf{Com}_{\beta'}(x_i^A; r_i^A) \end{array} \right\}. \tag{57}$$

Note that R uses $(1, x_1^A, r_1^A)$ as the witness when executing this WIPoK.

5. **(WIPoK-1-B.)** R and C execute an instance of WIPoK where R proves to C that he "knows" a pre-image of some y_i^B contained in Y^B, and y_i^B are consistent with com_i^B (as defined in Step 3b). Formally, R proves that $(Y^B, \{\mathsf{com}_i^B\}_{i\in[n-t]}) \in \mathcal{L}^{n-t}_{f,\beta'}$, where

$$\mathcal{L}^{n-t}_{f,\beta'} := \left\{ (y_1^B, \ldots, y_{n-t}^B), (\mathsf{com}_1^B, \ldots, \mathsf{com}_{n-t}^B) \;\middle|\; \exists (i, x_i^B, r_i^B) \text{ s.t. } \begin{array}{l} i \in [n-t] \wedge \\ y_i^B = f(x_i^B) \wedge \\ \mathsf{com}_i^B = \mathsf{Com}_{\beta'}(x_i^B; r_i^B) \end{array} \right\}. \tag{58}$$

Note that R uses $(1, x_1^B, r_1^B)$ as the witness when executing this WIPoK.

6. **(WIPoK-2.)** C and R execute an instance of WIPoK where C proves to R that he "knows" the message committed in com (defined in Step 2), *or* a pre-image of some y_i^A contained in Y^A (defined in Step 3a), *or* a pre-image of some y_i^B contained in Y^B (defined in Step 3b). Formally, C proves that $(\mathsf{com}, Y^A, Y^B) \in \mathcal{L}_\beta \vee \mathcal{L}_f^t \vee \mathcal{L}_f^{n-t}$, where

$$\mathcal{L}_\beta := \{\mathsf{com} \mid \exists (m, r) \text{ s.t. } \mathsf{com} = \mathsf{Com}_\beta(m; r)\}, \tag{59}$$
$$\mathcal{L}_f^t := \{(y_1^A, \ldots, y_t^A) \mid \exists (i, x_i^A) \text{ s.t. } i \in [t] \wedge y_i^A = f(x_i^A)\}, \tag{60}$$
$$\mathcal{L}_f^{n-t} := \{(y_1^B, \ldots, y_{n-t}^B) \mid \exists (i, x_i^B) \text{ s.t. } i \in [n-t] \wedge y_i^B = f(x_i^B)\}. \tag{61}$$

Note that C uses the (m, r) defined in Step 2 as the witness when executing this WIPoK.

Comment: Compared with Step 5 of Prot. 1, we add $\mathcal{L}_f^{n-t}$ as an additional OR part to the language to be proven. This is corresponding to the new **Slot-B** as discussed in Sec. 5.1.

Decommit Stage: C sends (m, r). R accepts if $\mathsf{com} = \mathsf{Com}_\beta(m; r)$, and rejects otherwise.

Security. Completeness is straightforward from the description of Prot. 2. The statistical binding property follows from that of Naor's commitment. Computational hiding of any commitment scheme follows directly from non-malleability. So it remains for us to show that our commitment protocol is non-malleable, which we establish by the following theorem.

Theorem 11. *The commitment scheme $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$ in Prot. 2 is non-malleable against synchronous PPT adversaries with tag space $[n]$, with n being any polynomial on λ.*

5.3 Proving Non-Malleability (Proof of Thm. 11)

Since we only consider synchronous adversaries in this section, we can assume that the adversary declares t and $\widetilde{t}$ at the beginning w.l.o.g. Thus, we can analyze the three cases of $t = \widetilde{t}$, $t < \widetilde{t}$, and $t > \widetilde{t}$ *separately*. Since the case of $t = \widetilde{t}$ is already ruled out by the definition of non-malleability, we prove non-malleability for the other two cases below.

5.3.1 The Case of $t < \widetilde{t}$

The proof for this case is almost identical to the proof of Prot. 1 in Sec. 4. To avoid re-doing the same work, we only focus on the places where modifications are needed. In the following, we assume that the reader has already read and understood the proof in Sec. 4.

First, by repeating the similar arguments in Sec. 4.2 to 4.4, we can see that it suffices to construct a machine $\mathcal{K}$ that satisfies similar properties as required in Lem. 9. Roughly speaking, $\mathcal{K}$ is required to extract the committed message $\widetilde{m}$ in the right session with a noticeable probability if it is given a prefix that leads to acceptance with a noticeable probability.[25] The construction of $\mathcal{K}$ is identical to that in Sec. 4.5 except that it extracts a witness (j, x_j^A, r_j^A) from **WIPoK-1-A** and always uses the same witness $(1, \widetilde{x}_1^B, \widetilde{r}_1^B)$ for the right **WIPoK-1-B**. Specifically, $\mathcal{K}$ honestly runs the man-in-the-middle experiment starting from the given prefix except for the following modifications:

1. Use the witness $(i, \widetilde{x}_i^A, \widetilde{r}_i^A)$ for the right **WIPoK-1-A** for $i \xleftarrow{\$} [\widetilde{t}]$;
2. Extract a witness (j, x_j^A, r_j^A) from the left **WIPoK-1-A**;
3. Use the witness (j, x_j^A) in the left **WIPoK-2**;
4. Extract a witness $\widetilde{w}$ from the right **WIPoK-2**;
5. If $\widetilde{w}$ is a valid witness $(\widetilde{m}, \widetilde{r})$ for $\widetilde{\mathsf{com}} \in \mathcal{L}_{\widetilde{\beta}}$, it outputs $\widetilde{m}$. Otherwise, it outputs $\bot$.

We stress again that $\mathcal{K}$ uses the same witness $(1, \widetilde{x}_1^B, \widetilde{r}_1^B)$ for the right **WIPoK-1-B** regardless of the i sampled in Item 1.[26] We also remark that $\mathcal{K}$ extracts a witness from the left **WIPoK-1-A** but not from the left **WIPoK-1-B**.

The proof that the above $\mathcal{K}$ satisfies similar properties as in Lem. 9 is almost identical to that in Sec. 4.5 to 4.7 in the one-sided setting. The only difference is the following. We define $\mathcal{K}_i$ similarly to that in Sec. 4.5, i.e., it works similarly to $\mathcal{K}$ with the random i (in Item 1) being fixed. We need to ensure that the witness $\widetilde{w}$ extracted from the right **WIPoK-2** by $\mathcal{K}_i$ cannot be a witness for **Slot-B**, i.e., a witness for $Y^B \in \mathcal{L}_f^{n-t}$ with a non-negligible probability. This is needed in the proof of a counterpart of Claim 8 in Sec. 4.6. It can be shown as follows:

- It is easy to see that $\widetilde{w} = (j, \widetilde{x}_j^B)$ for $j \neq 1$ with a negligible probability: Since $\mathcal{K}_i$ uses $(1, \widetilde{x}_1^B, \widetilde{r}_1^B)$ for the right **WIPoK-1-B**, extracting $(j, \widetilde{x}_j^B, \widetilde{r}_j^B)$ for $j \neq 1$ directly breaks the one-wayness of f or the computational hiding of Com.
- Suppose that $\widetilde{w} = (1, \widetilde{x}_1^B)$ with a non-negligible probability. The probability remains non-negligible even if we replace the witness used in the right **WIPoK-1-B** with $(2, \widetilde{x}_2^B, \widetilde{r}_2^B)$ by the WI property of the right **WIPoK-1-B**. Then, due to a similar reason to the above case, this also breaks the one-wayness of f or the computational hiding of Com. It is worth mentioning that this step relies on the fact that the left **WIPoK-1-A** and the right **WIPoK-1-B** do not interleave with each

[25] Strictly speaking, the input of $\mathcal{K}$ is not a prefix, but a prefix *without C's state* ST_C. We also remark that the assumption is not that the prefix leads to R's acceptance with a noticeable probability in the real experiment, but in the *simulated* experiment defined similarly to $\mathcal{G}_1$ in Sec. 4.4.

[26] Changing the witness for **Slot-B** to $(i, \widetilde{x}_i^B, \widetilde{r}_i^B)$ by using the WI property also works, but that is redundant.

other by the synchronicity assumption. (Otherwise we have to rewind the right **WIPoK-1-B** when rewinding the left **WIPoK-1-A**, in which case we cannot rely on the WI property of the right **WIPoK-1-B**.) This point will become relevant when dealing with asynchronous adversaries in Sec. 6.

Except for the above, the rest of the proof follows from a straightforward adaptation of that in Sec. 4.

5.3.2 The Case of $t > \widetilde{t}$

In this case, we have $n - t < n - \widetilde{t}$. Thus, we can simply do the same analysis as the case of $t < \widetilde{t}$ with the roles of **Slot-A** and **Slot-B** being swapped.

By combining the analysis for the above cases, we obtain Thm. 11.

6 Small-Tag, Asynchronous, Classical Setting

In this section, we show how to make $\langle C, R\rangle^{\mathsf{sync}}_{\mathsf{tg}}$ (i.e., Prot. 2) secure against asynchronous adversaries, to obtain the protocol $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg}}$. We emphasize that this section still focuses on the small-tag setting against classical (i.e., non-uniform PPT) adversaries.

6.1 High-Level Idea

By a careful inspection, the proof of non-malleability of Prot. 2 (which is based on the proof of non-malleability of Prot. 1) relies on the following assumptions about the schedule.

– The prefix of both the left and right sessions should be generated before moving forward. That is, the left (resp. right) message of Step 2 is sent before the right (resp. left) message of Step 3 is sent. This is necessary because we need to analyze the protocol for each fixed prefix pref in the security proof for Prot. 2.

– The left message of Step 3 is sent before the right **WIPoK-1-A** starts. This is needed because we rely on brute-force search to break hiding of com_i^A or com_i^B in an intermediate hybrid, where we rely on (non-uniform) WI of the right **WIPoK-1-A**. This is possible only if we can do brute-force before the right **WIPoK-1-A** starts so that the result of the brute-force can be treated as a non-uniform advice in the reduction to WI. (See Rmk. 12.)

– The left (resp. right) **WIPoK-1-A** and the right (resp. left) **WIPoK-1-B** do not interleave with each other. This is needed because we need to ensure that when rewinding the left **WIPoK-1-A** (resp. **WIPoK-1-B**), we do not rewind the right **WIPoK-1-B** (resp. **WIPoK-1-A**) as explained in Sec. 5.3.1.

– The left **WIPoK-1-B** finishes before the right **WIPoK-2** starts. This is needed because we rewind both of them simultaneously in the construction of the extractor $\mathcal{K}$.[27]

To deal with a schedule that does not satisfy some of the above conditions, we need to modify the protocol. Our modification is based on a combination of the following tricks.

1. Insert an extractable commitment to m by C between Steps X and $X+1$. We also modify the language for **WIPoK-2** to ensure that the committed message in the extractable commitment is the correct one. This gives us an opportunity to extract the right message $\widetilde{m}$ without extracting

[27] In fact, it is also fine that the left **WIPoK-1-A** *starts* after the right **WIPoK-2** *finishes*, since they do not interleave with each other in this case either.

the left message m in the man-in-the-middle game if the right Step $X+1$ starts before the left Step X finishes.

2. Insert a WIPoK proving some "trapdoor statement" where R plays the role of the prover between Steps X and $X+1$. Here, the trapdoor is a witness for an NP language chosen by R. We also modify the language for **WIPoK-2** to allow C to use the trapdoor as a witness. Now, if the left Step $X+1$ starts before the right Step X finishes, then we can extract the left trapdoor without extracting the right trapdoor in the man-in-the-middle game. Then, we can use the extracted trapdoor as a witness for the left **WIPoK-2**. As a result, we can simulate the honest committer in the left session without using the message m. This enables us to extract the right message $\widetilde{m}$ from the right **WIPoK-2** without using the left message m, and eventually to prove non-malleability.
3. Repeat Step X as many times as the total round complexity of all steps before Step X plus one. By the pigeonhole principle, this ensures that there is at least one execution of the left (resp. right) Step X that does not interleave with all steps before Step X in the right (resp. left). In particular, when we rewind that execution, it does not affect the security of any primitive on the other side.

6.2 Construction of $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$

We present our construction in Prot. 3 where constants $n_5, ..., n_9$ are specified later. This protocol makes use of the same building blocks as for Prot. 2 plus an extractable commitment, namely:

– An OWF f,

– Naor's commitment Com;

– A witness-indistinguishable proof of knowledge WIPoK (as per Def. 13);

– An extractable commitment ExtCom (as per Def. 8). We denote by $\mathsf{Verify}_{\mathsf{ExtCom}}$ the verification algorithm of ExtCom in the decommit stage.

Protocol 3: Small-Tag Asynchronous NMCom $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$

The tag space is defined to be $[n]$ where n is a polynomial on λ. Let $t \in [n]$ be the tag for the following interaction. Let m be the message to be committed to.

Commit Stage:

1. Receiver R samples and sends the first message β for Naor's commitment.
2. Committer C commits to m using the second message of Naor's commitment. C also sends a β' that will be used as the first Naor's message for R to generate a commitment in next step. Formally, C samples r and β' and sends the tuple $\big(\mathsf{com} = \mathsf{Com}_\beta(m; r),\ \beta'\big)$.
3. R and C do the following:
   (a) **(TrapGen.)** R computes $V_0 = f(v_0)$ and $V_1 = f(v_1)$ with $v_0, v_1 \xleftarrow{\$} \{0,1\}^\lambda$ and sends (V_0, V_1) to C
   (b) **(WIPoK-Trap.)** R and C execute an instance of WIPoK where R proves to C that he "knows" a pre-image of V_0 or V_1. Formally, R proves that $(V_0, V_1) \in \mathcal{L}_f^{\mathrm{OR}}$, where

$$\mathcal{L}_f^{\mathrm{OR}} := \big\{(V_0, V_1) \bigm| \exists v \ s.t. \ \ f(v) = V_0 \ \vee \ f(v) = V_1\big\}. \tag{62}$$

   Note that R uses v_0 as the witness when executing this WIPoK.

Comment: This step is inserted for Trick 2 in Sec. 6.1 where a witness for $(V_0, V_1) \in \mathcal{L}_f^{\mathrm{OR}}$ plays the role of a "trapdoor".

4. R performs the following computation:

   (a) R computes $\{y_i^A = f(x_i^A)\}_{i\in[t]}$ with $x_i^A \xleftarrow{\$} \{0,1\}^\lambda$ for each $i \in [t]$, and sets $Y^A := (y_1^A, \ldots, y_t^A)$. R also computes $\{\mathsf{com}_i^A = \mathsf{Com}_{\beta'}(x_i^A; r_i^A)\}_{i\in[t]}$, where r_i^A is a random string for each $i \in [t]$.

   (b) R computes $\{y_i^B = f(x_i^B)\}_{i\in[n-t]}$ with $x_i^B \xleftarrow{\$} \{0,1\}^\lambda$ for each $i \in [n-t]$, and sets $Y^B := (y_1^B, \ldots, y_{n-t}^B)$. R also computes $\{\mathsf{com}_i^B = \mathsf{Com}_{\beta'}(x_i^B; r_i^B)\}_{i\in[n-t]}$, where r_i^B is a random string for each $i \in [n-t]$.

   R sends the tuple $\big(Y^A,\ \{\mathsf{com}_i^A = \mathsf{Com}_{\beta'}(x_i^A; r_i^A)\}_{i\in[t]},\ Y^B,\ \{\mathsf{com}_i^B = \mathsf{Com}_{\beta'}(x_i^B; r_i^B)\}_{i\in[n-t]}\big)$.

5. **(ExtCom-1 $\times$ n_5.)** C and R sequentially execute n_5 instances of ExtCom where C commits to m. For $i \in [n_5]$, let τ_i^1 and decom_i^1 be the transcript and decommitment information (privately obtained by C) of the i-th execution.

   Comment: This step is inserted for Trick 1 and repeated for Trick 3 in Sec. 6.1.

6. **(WIPoK-1-A $\times$ n_6.)** R and C sequentially execute n_6 instances of WIPoK where R proves to C that he "knows" a pre-image of some y_i^A contained in Y^A, and y_i^A are consistent with com_i^A (as defined in Step 4a). Formally, R proves that $(Y^A, \{\mathsf{com}_i^A\}_{i\in[t]}) \in \mathcal{L}_{f,\beta'}^t$, where

$$\mathcal{L}_{f,\beta'}^t := \left\{ (y_1^A, \ldots, y_t^A), (\mathsf{com}_1^A, \ldots, \mathsf{com}_t^A) \;\middle|\; \exists (i, x_i^A, r_i^A)\ s.t.\ \begin{array}{l} i \in [t]\ \wedge \\ y_i^A = f(x_i^A)\ \wedge \\ \mathsf{com}_i^A = \mathsf{Com}_{\beta'}(x_i^A; r_i^A) \end{array} \right\}. \quad (63)$$

   Note that R uses $(1, x_1^A, r_1^A)$ as the witness when executing this WIPoK.

   Comment: This step is repeated for Trick 3 in Sec. 6.1.

7. **(ExtCom-2 $\times$ n_7.)** C and R sequentially execute n_7 instances of ExtCom where C commits to m. Let τ_i^2 and decom_i^2 be the transcript and decommitment (privately obtained by C) of the i-th execution.

   Comment: This step is inserted for Trick 1 and repeated for Trick 3 in Sec. 6.1.

8. **(WIPoK-1-B $\times$ n_8.)** R and C sequentially execute n_8 instances of WIPoK where R proves to C that he "knows" a pre-image of some y_i^B contained in Y^B, and y_i^B are consistent with com_i^B (as defined in Step 4b). Formally, R proves that $(Y^B, \{\mathsf{com}_i^B\}_{i\in[n-t]}) \in \mathcal{L}_{f,\beta'}^{n-t}$, where

$$\mathcal{L}_{f,\beta'}^{n-t} := \left\{ (y_1^B, \ldots, y_{n-t}^B), (\mathsf{com}_1^B, \ldots, \mathsf{com}_{n-t}^B) \;\middle|\; \exists (i, x_i^B, r_i^B)\ s.t.\ \begin{array}{l} i \in [n-t]\ \wedge \\ y_i^B = f(x_i^B)\ \wedge \\ \mathsf{com}_i^B = \mathsf{Com}_{\beta'}(x_i^B; r_i^B) \end{array} \right\}. \quad (64)$$

   Note that R uses $(1, x_1^B, r_1^B)$ as the witness when executing this WIPoK.

   Comment: This step is repeated for Trick 3 in Sec. 6.1.

9. **(ExtCom-3 $\times$ n_9.)** C and R sequentially execute n_9 instances of ExtCom where C commits to m. Let τ_i^2 and decom_i^2 be the transcript and decommitment (privately obtained by C) of the i-th execution.

   Comment: This step is inserted for Trick 1 and repeated for Trick 3 in Sec. 6.1.

10. **(WIPoK-2.)** C and R execute an instance of WIPoK where C proves to R that he "knows" the (same) message committed in com (defined in Step 2) and all commitments τ_i^c of ExtCom (in Steps 5, 7 and 9), *or* a pre-image of some y_i^A contained in Y^A (defined in Step 4a), *or* a pre-image of some y_i^B contained in Y^B (defined in Step 4b) *or* a preimage of either of V_0 or V_1 (defined in Step 3a). Formally, C proves that $(\mathsf{com}, \{\tau_i^c\}_{(c,i)\in J}, Y^A, Y^B, V_0, V_1) \in \mathcal{L}_\beta \vee \mathcal{L}_f^t \vee \mathcal{L}_f^{n-t} \vee \mathcal{L}_f^{\mathrm{OR}}$, where $J = (\{1\} \times [n_5]) \cup (\{2\} \times [n_7]) \cup (\{3\} \times [n_9])$,

$$\mathcal{L}_\beta := \left\{ (\mathsf{com}, \{\tau_i^c\}_{(c,i)\in J}) \;\middle|\; \begin{array}{l} \exists (m, r, \{\mathsf{decom}_i^c\}_{(c,i)\in J}) \textit{ s.t.} \\ \mathsf{com} = \mathsf{Com}_\beta(m; r) \;\wedge \\ \forall\ (c,i) \in J,\ \mathsf{Verify}_{\mathsf{ExtCom}}(m, \tau_i^c, \mathsf{decom}_i^c) = \top \end{array} \right\} \tag{65}$$

$$\mathcal{L}_f^t := \{(y_1^A, \ldots, y_t^A) \mid \exists (i, x_i^A) \textit{ s.t. } i \in [t] \wedge y_i^A = f(x_i^A)\}, \tag{66}$$

$$\mathcal{L}_f^{n-t} := \{(y_1^B, \ldots, y_{n-t}^B) \mid \exists (i, x_i^B) \textit{ s.t. } i \in [n-t] \wedge y_i^B = f(x_i^B)\}, \tag{67}$$

and $\mathcal{L}_f^{\mathrm{OR}}$ is defined in Language (62). Note that C uses the $(m, r, \{\mathsf{decom}_i^c\}_{(c,i)\in J})$ defined in Steps 2, 5, 7 and 9 as the witness when executing this WIPoK.

<u>Comment:</u> Compared with Step 6 of Prot. 2, we add $\mathcal{L}_f^{\mathrm{OR}}$ as an additional OR part to the language to be proven for Trick 2 in Sec. 6.1. We also modify $\mathcal{L}_\beta$ to prove that all commitments τ_i^c of ExtCom commit to the same message as com for Trick 1 in Sec. 6.1.

Decommit Stage: C sends (m, r). R accepts if $\mathsf{com} = \mathsf{Com}_\beta(m; r)$, and rejects otherwise.

Choice of n_i. We recursively define $n_5, ..., n_9$ so that n_i is larger than the total round complexity of Steps 1 to $i-1$. We also require that n_5 is larger than the round complexities of WIPoK and ExtCom. It is easy to see that we can set n_i to be constant if both WIPoK and ExtCom have constant rounds. In particular, Prot. 3 runs in constant rounds. We remark that the above choice of n_i is for simplifying the security analysis and we could significantly optimize the exact round complexity with more careful analysis. But we choose not to press the issue further.

Notation. For $c = 1, 2, 3$, we refer to the i-th execution of ExtCom in **ExtCom-c** as **ExtCom-c-i**. We also define **WIPoK-1-A-i** and **WIPoK-1-B-i** similarly.

Security. The completeness is easy to see. Statistical binding follows from that of Naor's commitment. We prove computational hiding below.[28]

Theorem 12. *The commitment scheme $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$ in Prot. 3 is computationally hiding.*

Proof. Let R^* be a non-uniform PPT adversary against the computational hiding property of Prot. 3. We consider the following sequence of hybrids.

– Hybrid $H_1^{R^*}(\lambda, m)$: This is the real experiment that simulates the interaction between $C(m)$ and R^*. That is, it runs $\langle C(m), R^*\rangle_{\mathsf{tg}}(1^\lambda)$ and outputs the final output of R^*. We have to prove

$$\left| \Pr\left[H_1^{R^*}(\lambda, m_0) = 1\right] - \Pr\left[H_1^{R^*}(\lambda, m_1) = 1\right] \right| = \mathsf{negl}(\lambda). \tag{68}$$

for all m_0, m_1.

[28] Though computational hiding follows from non-malleability, we prove it here because we rely on the computational hiding in the proof of non-malleability.

– Hybrid $H_2^{R^*}(\lambda, m)$: This is identical to $H_1^{R^*}(\lambda, m)$ except that it runs the emulation extractor for **WIPoK-1-A-1** to extract a witness (i, x_i^A, r_i^A) for $(Y^A, \{\mathsf{com}_i^A\}_{i\in[t]}) \in \mathcal{L}^t_{f,\beta'}$ where Y^A and $\{\mathsf{com}_i^A\}_{i\in[t]})$ are generated in Step 4a. We note that this experiment extracts (i, x_i^A, r_i^A) but does not use it. By the PoK property (as per Def. 12) of **WIPoK-1-A-1**, we have

$$\left|\Pr\left[H_1^{R^*}(\lambda, m) = 1\right] - \Pr\left[H_2^{R^*}(\lambda, m) = 1\right]\right| = \mathsf{negl}(\lambda) \tag{69}$$

for all m.

– Hybrid $H_3^{R^*}(\lambda, m)$: This is identical to $H_2^{R^*}(\lambda, m)$ except that it uses (i, x_i^A, r_i^A) extracted from **WIPoK-1-A-1** in **WIPoK-2**. By the WI property of **WIPoK-2**, we have

$$\left|\Pr\left[H_2^{R^*}(\lambda, m) = 1\right] - \Pr\left[H_3^{R^*}(\lambda, m) = 1\right]\right| = \mathsf{negl}(\lambda) \tag{70}$$

for all m.
We observe that $H_3^{R^*}(\lambda, m_b)$ uses m only in the generation of com (in Step 2) and τ_i^c for $(c, i) \in J$ (in Steps 5, 7 and 9). Therefore, by the computational hiding property of Naor's commitment and ExtCom, we have

$$\left|\Pr\left[H_3^{R^*}(\lambda, m_0) = 1\right] - \Pr\left[H_3^{R^*}(\lambda, m_1) = 1\right]\right| = \mathsf{negl}(\lambda) \tag{71}$$

for all m_0, m_1. Combining Eq. (69) to (71), we obtain Eq. (68).

This completes the proof of Thm. 12.

□

We establish non-malleability by Thm. 13, whose proof will be presented in Sec. 6.4.

Theorem 13. *The commitment scheme $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$ in Prot. 3 is non-malleable against asynchronous PPT adversaries with tag space $[n]$, with n being any polynomial on λ.*

6.3 Adaptive Schedule vs. Predetermined Schedule

Before proving the non-malleability of Prot. 3, we prove a useful lemma that enable us to use *predetermined* schedules when proving non-malleability. The following Lem. 11 should be attributed to [BLS22, Lemma 5.1]: Although they only show this lemma for their specific construction, their proof technique is general enough. Our proof of Lem. 11 is identical to theirs except for some notational changes customized to our application.

Lemma 11 ([BLS22, Lemma 5.1]). *Let $\mathcal{S}$ be the set of all possible schedules of a MIM adversary against a commitment scheme $\langle C, R\rangle$. Let $S_1, ..., S_N$ be efficiently recognizable subsets of $\mathcal{S}$ such that $\bigcup_{i\in[N]} S_i = \mathcal{S}$ for $N = \mathsf{poly}(\lambda)$. If* for every *$i \in [N]$, $\langle C, R\rangle$ is non-malleable against non-uniform PPT MIM adversaries whose schedule belongs to S_i, then it is also non-malleable against arbitrary non-uniform PPT MIM adversaries.*

Proof. We can assume that $S_1, ..., S_N$ are disjoint w.l.o.g. (If they are not disjoint, we can define $S_1' := S_1$, $S_i' := S_i \setminus \left(\bigcup_{j\in[i-1]} S_j\right)$ for $i > 1$ and consider disjoint sets $S_1', ..., S_N'$.)

Given an arbitrary non-uniform PPT MIM $\mathcal{M}$ and non-uniform PPT distinguisher $\mathcal{D}$ that break non-malleability for some values m_0, m_1 with advantage δ, we construct a new non-uniform PPT adversary with a schedule that belongs to S_i for a predetermined $i \in [n]$, which breaks the scheme with probability δ/N.

Consider an adversary $\mathcal{M}'$ that first samples uniformly at random $i \xleftarrow{\$} [N]$. $\mathcal{M}'$ then runs $\mathcal{M}$ to complete the MIM attack. If the transcript generated in the execution belongs to S_i, $\mathcal{M}'$ outputs whatever $\mathcal{M}$ outputs, and otherwise outputs $\perp$.

Since every execution must belong to exactly one of $S_1, ..., S_N$, $\mathcal{M}'$ breaks non-malleability with probability exactly δ/N (with respect to the same distinguisher $\mathcal{D}$ and m_0, m_1). Finally, by an averaging argument, we fix the choice of $\mathcal{M}'$ for a schedule to be of some type S_i that maximizes $\mathcal{D}$'s distinguishing advantage. We obtain a corresponding MIM with a schedule that belongs to S_i for a predetermined $i \in [n]$ with at least the same advantage δ/N.

□

6.4 Proving Non-Malleability (Proof of Thm. 13)

For proving Thm. 13, we consider the following "bad" schedules:

- Bad 1: The right message of Step 4 is sent before the left message of Step 2 is sent.
- Bad 2: The left message of Step 4 is sent before the right message of Step 2 is sent.
- Bad 3: The right **WIPoK-1-A** starts before the left message of Step 4 is sent.
- Bad 4: The right **WIPoK-1-B** starts before the left **WIPoK-1-A** finishes.
- Bad 5: The right **WIPoK-2** starts before the left **WIPoK-1-B** finishes.

In the above, when we say "**WIPoK-XX** starts", this means that the first message of the first execution of WIPoK in **WIPoK-XX** is sent. Similarly, when we say "**WIPoK-XX** finishes", this means that the final message of the final execution of WIPoK in **WIPoK-XX** is sent. We remark that the above bad cases are not disjoint.

We first prove non-malleability in the case where the schedule does not suffer from any of them (which we call a *good* schedule) in Sec. 6.5. Then, we prove it for the remaining schedules (which we call *bad* schedules) in Sec. 6.6. We remark that we can analyze them separately due to Lem. 11.

6.5 Non-Malleability for Good Schedules

Based on the observations made in Sec. 6.1, we can show the non-malleability analogously to the synchronous case if none of Bad 1-5 occurs.

Lemma 12. *The commitment scheme $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$ in Prot. 3 is non-malleable against asynchronous PPT adversaries whose schedule does not suffer from any of* Bad 1-5.

Proof. Similarly to the proof of non-malleability of Prot. 2 in Sec. 5, we consider the three cases of $t = \widetilde{t}$, $t < \widetilde{t}$, and $\widetilde{t} < t$ separately where t and $\widetilde{t}$ are the tags of the left and right sessions, respectively. We remark that we can assume that the reduction algorithm knows which of them happens from the beginning because of Lem. 11, even if the adversary adaptively determines the schedule.

Again, the case of $t = \widetilde{t}$ is already ruled out by the definition of non-malleability. Next, we analyze the remaining two cases in Sec. 6.5.1 and 6.5.2, which will eventually complete the proof of Lem. 12.

□

6.5.1 The Case of $t < \widetilde{t}$.

For a PPT man-in-the-middle adversary $\mathcal{M}$ with non-uniform advice z and a message m, we define the prefix generation algorithm $H_{\mathsf{pre}}^{\mathcal{M}}(\lambda, m, z)$ as follows similarly to that in Algo. 4.1: $H_{\mathsf{pre}}^{\mathcal{M}}(\lambda, m, z)$ runs the man-in-the-middle experiment $\mathsf{mim}_{\langle C,R\rangle_{\mathsf{tg}}^{\mathsf{async}}}^{\mathcal{M}}(\lambda, m, z)$ until $\mathcal{M}$ receives the left message of

Step 2 *and* sends the right message of Step 2. Then, it outputs the prefix $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ where $\mathsf{st}_{\mathcal{M}}$, st_C, st_R are the states of $\mathcal{M}$, C, and R at this point, respectively; And τ (reps. $\widetilde{\tau}$) is Naor's commitment in the right (resp. left) session generated in Steps 1 and 2. We denote by $\mathcal{A}(\mathsf{st}_{\mathcal{M}})$ to mean the algorithm that works similarly to $\mathcal{M}$ from the point where the prefix is generated. We remark that $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$ does not reach to Step 4 in either of the left or right session since we assume that neither of $\mathsf{Bad}\ 1$ or $\mathsf{Bad}\ 2$ occurs.

We define a machine $\mathcal{G}_i$ similarly to that used in the proof of Lem. 9 (recall it from Fig. 6) except that C commits to 0 instead of m in the left session. Formally, it works as follows:

Machine $\mathcal{G}_i$: For a prefix $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$, $\mathcal{G}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ runs the rest of $\mathsf{mim}^{\mathcal{M}}_{\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg}}}(\lambda, m, z)$ except for the following three differences:

1. C commits to 0 instead of m in the left **ExtCom-{1,2,3}**.
2. R uses $(i, \widetilde{x}^A_i)$ instead of $(1, \widetilde{x}^A_1)$ in the right **WIPoK-1-A**.
3. It chooses i^* such that the left **WIPoK-1-A-i^*** does not interleave with the right Step 1 to 5, i.e., no message of the right Step 1 to 5 is sent during the execution of the left **WIPoK-1-A-i^***. Note that such i^* exists by the pigeonhole principle, and we call the smallest such i^* the *good* index. Then, instead of executing the left **WIPoK-1-A-i^*** honestly, it uses the witness-extended emulator $\mathcal{WE}$ to extract a witness:
   - If the left committer accepts the left **WIPoK-1-A-i^*** and the extracted witness (j, x^A_j, r^A_j) is valid, $\mathcal{G}_i$ uses (j, x^A_j) to finish the left **WIPoK-2** and outputs $\mathcal{M}$'s final state and the right receiver's decision bit b;
   - If the left committer accepts the left **WIPoK-1-A-i^*** but the extracted witness is invalid, it aborts immediately and outputs $(\bot, \bot)$;
   - If the left committer rejects the left **WIPoK-1-A-i^***, it runs the rest of man-in-the-middle experiment to output $\mathcal{M}$'s final state and the right receiver's decision bit b. Note that it does not need to run the left **WIPoK-2** in this case since the left committer aborts after the left **WIPoK-1-A-i^***.

We let $p^{\mathsf{Sim}}_{\mathsf{pref}}$ be the probability that $\mathcal{G}_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ returns $b = \top$.

Remark 14. Strictly speaking, the above description of $\mathcal{G}_i$ is not well-defined since the good index depends on the schedule, which is adaptively determined. However, since there are constant number of possibilities for the good index, we only have to prove non-malleability assuming that the good index is fixed at first by Lem. 11. A similar remark applies to later parts of the current proof. We will not repeat it as it should be clear from context.

By repeating similar arguments as in Sec. 4.2 to 4.4, we can see that it suffices to prove the following lemma, which is an analog of Lem. 9.

Lemma 13. *There exists an expected PPT machine $\mathcal{K}$ such that for any $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ in the support of $H^{\mathcal{M}}_{\mathsf{pre}}(\lambda, m, z)$ and any noticeable $\varepsilon(\lambda)$, the following holds:*

1. ***(Syntax.)*** *$\mathcal{K}$ takes as input $(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$ and makes oracle access to $\mathcal{A}(\mathsf{st}_{\mathcal{M}})$. It output a value $\mathsf{Val}_{\mathcal{K}} \in \{0,1\}^{\ell(\lambda)} \cup \{\bot\}$ such that $\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau})$ whenever $\mathsf{Val}_{\mathcal{K}} \neq \bot$.*
2. *If $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$, then it holds that*

$$\Pr\Big[\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val}_{\mathcal{K}} \leftarrow \mathcal{K}^{\mathcal{A}(\mathsf{st}_{\mathcal{M}})}(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})\Big] \geq \frac{\varepsilon'(\lambda)}{\widetilde{t}},$$

where $\varepsilon'(\lambda) := \frac{\varepsilon(\lambda)}{10t^2}$.

Proof of Lem. 13. We construct $\mathcal{K}_i$ and $\mathcal{K}$ similarly as in Sec. 5.3.1, which is in turn based on the technique in Sec. 4.5.

Machine $\mathcal{K}_i$ ($i \in [\widetilde{t}]$): On input $(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$, it behaves identically to $\mathcal{G}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except for the following difference. Machine $\mathcal{K}_i^{\mathcal{A}(\mathsf{st}_\mathcal{M})}(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau}))$ uses the witness-extended emulator $\mathcal{WE}$ to finish the right **WIPoK-2**, instead of playing the role of the honest receiver. If it extracts a witness for $(\widetilde{\mathsf{com}}, \{\widetilde{\tau}_i^c\}_{(c,i)\in J}) \in \mathcal{L}_{\widetilde{\beta}}$, then it sets Val to be the first component $\widetilde{m}$ of the witness (which is suppose to be the committed message in $\widetilde{\tau}$) and outputs $\mathsf{Val} = \widetilde{m}$; Otherwise, it outputs $\bot$.

Extractor $\mathcal{K}$: On input $(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$, it samples uniformly at random an index $i \xleftarrow{\$} [\widetilde{t}]$, executes $\mathcal{K}_i^{\mathcal{A}(\mathsf{st}_\mathcal{M})}(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$, and outputs whatever $\mathcal{K}_i^{\mathcal{A}(\mathsf{st}_\mathcal{M})}(1^\lambda, \mathsf{st}_R, \tau, \widetilde{\tau})$ outputs.

We can show that $\mathcal{K}$ as described above satisfies Lem. 13 by similar analyses to those in Sec. 4.5 to 4.7 additionally relying on the computational hiding property of ExtCom along with the following observations:

1. By our choice of the good index i^*, the witness-extended emulator for the left **WIPoK-1-A-i^*** does not rewind the right **WIPoK-Trap**. Thus, $\mathcal{K}_i$ does not extract a "trapdoor witness" (i.e., a witness for $(\widetilde{V}_0, \widetilde{V}_1) \in \mathcal{L}_f^{\mathrm{OR}}$) except for a negligible probability by the one-wayness of f and the WI property of the right **WIPoK-Trap**.
2. Since we assume that Bad 4 does not occur, the witness-extended emulator for the left **WIPoK-1-A-i^*** does not rewind the right **WIPoK-1-B**. Thus, $\mathcal{K}_i$ does not extract a witness for **Slot-B**, (i.e., a witness for $\widetilde{Y}^B \in \mathcal{L}_f^{n-t}$) except for a negligible probability by the one-wayness of f and the WI property of the right **WIPoK-1-B**. This can be proven by a similar analysis to that in Sec. 5.3.1.
3. Since we assume that Bad 3 does not occur, we can rely on the WI property of the right **WIPoK-1-A** in a hybrid where we find the committed messages in $\{\mathsf{com}_i^A\}_{i\in[t]}$ by brute-force. This is needed in a step corresponding to the proof of Inequality (36) in Sec. 4.6.
4. Since we assume that Bad 5 does not occur, the left **WIPoK-1-A** and the right **WIPoK-2** do not run simultaneously. Thus, there is no nesting of extractors in hybrids where we invoke extractors for both of them (e.g., $\mathcal{K}$ and $\mathcal{K}_i$).

Remark 15. One might wonder how we rely on the computational hiding property of the left **ExtCom-{1,2,3}** even though the witness-extended emulator of the right **WIPoK-2** may rewind them. However, one can see by a close inspection of the proof that we only need to use computational hiding of the left **ExtCom-{1,2,3}** in hybrids that do not run the witness-extended emulator of the right **WIPoK-2**. This is similar to how we replace the witness used in the left **WIPoK-2** using the WI property in the proof for the one-sided synchronous case in Sec. 4.

This completes the proof of Lem. 13, which finishes the proof for the case of $t < \widetilde{t}$.

□

6.5.2 The Case of $t > \widetilde{t}$

Similarly to the synchronous case (i.e., the proof of Thm. 11), the proof of this case can be done similarly to that of the case of $t < \widetilde{t}$ except that we extract a witness from (the good index of) the left **WIPoK-1-B** instead of from **WIPoK-1-A**. In doing so, one has to be careful that Bad 4 is not symmetric for **Slot-A** and **Slot-B**. In particular, even if we assume that Bad 4 does not occur, it may be still possible that the *left* **WIPoK-1-B** and the *right* **WIPoK-1-A** run concurrently. However, this is not an issue since we rewind the left **WIPoK-1-B** of the good index that does not interleave with *any* message of the right session before **WIPoK-1-B**. In particular, its rewinding does not rewind *any* slot of the right **WIPoK-1-A**. Except for the above remark, the proof of this case is almost identical to the proof for the case of $t < \widetilde{t}$ with the roles of **Slot-A** and **Slot-B** being swapped.

6.6 Non-Malleability for Bad Schedules

Before analyzing bad cases, we show a lemma that is used many times in the analysis of bad cases.

Lemma 14. *Let C^* be a PPT cheating prover of the protocol in Prot. 3 with advice z. For $(c,i) \in J$ (where $J = (\{1\} \times [n_5]) \cup (\{2\} \times [n_7]) \cup (\{3\} \times [n_9])$ as defined in Step 10 of Prot. 3) we define experiments Exp and $\mathsf{Exp}_{c,i}$ as follows.*

– Experiment $\mathsf{Exp}(\lambda, z)$*: It simulates the interaction between C^* and R in the commit stage and outputs $(\mathsf{OUT}_{C^*}, \mathsf{val}_b(\tau))$ where OUT_{C^*} is the final output of C^*, b is R's decision bit, τ is Naor's commitment generated in Steps 1 and 2, and*

$$\mathsf{val}_b(\tau) := \begin{cases} \mathsf{val}(\tau) & b = \top \\ \bot & b = \bot \end{cases}.$$

– Experiment $\mathsf{Exp}_{c,i}(\lambda, z)$*: This experiment works similarly to $\mathsf{Exp}(\lambda, z)$ except that it runs the extractor of* **ExtCom-c-i** *to obtain the extracted message m^* and outputs $(\mathsf{OUT}_{C^*}, \mathit{\Gamma}_b(m^*))$ where*

$$\mathit{\Gamma}_b(m^*) := \begin{cases} m^* & b = \top \\ \bot & b = \bot \end{cases}.$$

Then, for all $(c,i) \in J$, it holds that

$$\{\mathsf{Exp}(\lambda, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^*} \overset{s}{\approx} \{\mathsf{Exp}_{c,i}(\lambda, z)\}_{\lambda \in \mathbb{N}, z \in \{0,1\}^*}.$$

Proof. First, we prove the following claim.

Claim 14.

$$\Pr\left[b = \top \wedge (\mathsf{com}, \{\tau_i^c\}_{(c,i)\in J}) \notin \mathcal{L}_\beta\right] = \mathsf{negl}(\lambda) \tag{72}$$

where the probability is taken over the execution of $\mathsf{Exp}(\lambda, z)$*.*

Proof of Claim 14. We consider a modified experiment $\mathsf{Exp}'(\lambda, z)$ that works similarly to $\mathsf{Exp}(\lambda, z)$ except that it runs the witness-extended emulator for **WIPoK-2** instead of running it honestly. By the PoK property (as per Def. 12), it suffices to prove Eq. (72) holds in $\mathsf{Exp}'(\lambda, z)$.

Let $\widetilde{w}$ be the extracted witness from **WIPoK-2**. By the PoK property (as per Def. 12), $\widetilde{w}$ is a valid witness for $(\mathsf{com}, \{\tau_i^c\}_{(c,i)\in J}, Y^A, Y^B, V_0, V_1) \in \mathcal{L}_\beta \vee \mathcal{L}_f^t \vee \mathcal{L}_f^{n-t} \vee \mathcal{L}_f^{\mathrm{OR}}$ except for a negligible probability when $b = \top$.

We can show that $\widetilde{w}$ cannot be a witness of $Y^A \in \mathcal{L}_f^t$ except for a negligible probability by using the one-wayness of f and WI of **WIPoK-1-A**. Indeed, this can be seen as follows.

– In $\mathsf{Exp}'(\lambda, z)$, the receiver uses $(1, x_1^A, r_1^A)$ as a witness for **WIPoK-1-A**. Therefore, if the probability that $\widetilde{w} = (j, x_j^A)$ such that $y_j^A = f(x_j^A)$ for some $j \neq 1$ is non-negligible, it clearly breaks the one-wayness of f or the computational hiding property of Com.

– If the probability that $\widetilde{w} = (1, x_1^A)$ such that $y_1^A = f(x_1^A)$ is non-negligible, by the WI property of **WIPoK-1-A**, a similar statement holds even if the receiver uses $(2, x_2^A, r_2^A)$ instead of $(1, x_1^A, r_1^A)$ as a witness for **WIPoK-1-A**. Then, by a similar argument to the above case, it also breaks the one-wayness of f or the computational hiding property of Com.

By similar arguments, $\widetilde{w}$ cannot be a witness of $Y^B \in \mathcal{L}_f^{n-t}$ or $(V_0, V_1) \in \mathcal{L}_f^{\mathrm{OR}}$ except for a negligible probability. Thus, $\widetilde{w}$ is a witness for $(\mathsf{com}, \{\tau_i^c\}_{(c,i)\in J}) \in \mathcal{L}_\beta$ except for a negligible probability when $b = \top$. This completes the proof of Claim 14.

□

Claim 14 in particular means that we have $\mathsf{val}(\tau) = \mathsf{val}(\tau_c^i) \neq \bot$ whenever $b = \top$ except for a negligible probability. Then, the extractability of ExtCom immediately implies Lem. 14.

□

Next, we analyze the bad cases one by one.

6.6.1 Analysis of Bad 1

Lemma 15. *The commitment scheme $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$ in Prot. 3 is non-malleable against asynchronous PPT adversaries whose schedule satisfies* Bad 1.

Proof. The proof for this case is easy. When Bad 1 occurs, the right commitment $\widetilde{\mathsf{com}}$ (in Step 2) is sent before the left commitment com (in Step 2) is sent. Then, a straightforward reduction to the computational hiding of the left session works by extracting $\widetilde{m}$ from $\widetilde{\mathsf{com}}$ by brute-force and treating it as non-uniform advice of an adversary against computational hiding.

□

6.6.2 Analysis of Bad 2

Lemma 16. *The commitment scheme $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$ in Prot. 3 is non-malleable against asynchronous PPT adversaries whose schedule satisfies* Bad 2.

Proof. Recall that Bad 2 means that the left message of Step 4 is sent before the right message of Step 2 is sent. We assume that $\mathcal{M}$ receives the right message of Step 1 at first w.l.o.g. In particular, no massage in the right session is sent during the execution of the left **WIPoK-Trap**.

We consider the following sequence of hybrids.

– Hybrid $H_1^{\mathcal{M}}(\lambda, m, z)$: This hybrid is the real man-in-the-middle experiment $\mathsf{mim}_{\langle C,R\rangle}^{\mathcal{M}}(\lambda, m, z)$. The output of the experiment can be written as $(\mathsf{OUT}_{\mathcal{M}}, \mathsf{val}_b(\widetilde{\tau}))$ where $\mathsf{OUT}_{\mathcal{M}}$ is $\mathcal{M}$'s final output, b is the R's decision bit, and $\widetilde{\tau}$ is Naor's commitment generated in Steps 1 and 2 in the right session, and

$$\mathsf{val}_b(\widetilde{\tau}) := \begin{cases} \mathsf{val}(\widetilde{\tau}) & b = \top \\ \bot & b = \bot \end{cases}.$$

– Hybrid $H_2^{\mathcal{M}}(\lambda, m, z)$: It is identical to $H_1^{\mathcal{M}}(\lambda, m, z)$ except that it runs the witness-extended emulator for the left **WIPoK-Trap** instead of running it honestly. Let v be the extracted witness. If C accepts the left **WIPoK-Trap** but v is not a valid witness for $(V_0, V_1) \in \mathcal{L}_f^{\mathrm{OR}}$, it aborts with outputting $(\bot, \bot)$. Otherwise, it works similarly to $H_1^{\mathcal{M}}(\lambda, m, z)$. We remark that it does not use v though it extracts it.

– Hybrid $H_3^{\mathcal{M}}(\lambda, m, z)$: It is identical to $H_2^{\mathcal{M}}(\lambda, m, z)$ except that it uses v extracted from the left **WIPoK-Trap** as a witness in the left **WIPoK-2**.

– Hybrid $H_4^{\mathcal{M}}(\lambda, m, z)$: It is identical to $H_3^{\mathcal{M}}(\lambda, m, z)$ except that it commits to 0 instead of m in all slots of **ExtCom-{1,2,3}**.

We prove that each pair of neighboring hybrids is computationally indistinguishable.

Claim 15. *It holds that*

$$\{H_1^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*} \overset{s}{\approx} \{H_2^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*}.$$

Proof of Claim 15. This follows from the PoK property (as per Def. 12) of the left **WIPoK-Trap** straightforwardly.

□

Claim 16. *It holds that*

$$\{H_2^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*} \overset{c}{\approx} \{H_3^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*}.$$

Proof of Claim 16. We introduce the following additional hybrids:

– Hybrid $\overline{H}_2^{\mathcal{M}}(\lambda, m, z)$: Choose the smallest $i^* \in [n_5]$ such that the right **ExtCom-1-i^*** does not interleave with the left **WIPoK-2**, i.e., no message of the left **WIPoK-2** is sent during the execution of the right **ExtCom-1-i^***.[29] Such i^* exists by the pigeonhole principle since we assume that n_5 is larger than the round complexity of **WIPoK-2**. This experiment is identical to $H_2^{\mathcal{M}}(\lambda, m, z)$ except that it runs the extractor of the right **ExtCom-1-i^*** to extract $\widetilde{m}^*$, and outputs $(\mathsf{OUT}_{\mathcal{M}}, \mathit{\Gamma}_b(\widetilde{m}^*))$ instead of $(\mathsf{OUT}_{\mathcal{M}}, \mathsf{val}_b(\widetilde{\tau}))$ where

$$\mathit{\Gamma}_b(m^*) := \begin{cases} m^* & b = \top \\ \bot & b = \bot \end{cases}.$$

– Hybrid $\overline{H}_3^{\mathcal{M}}(\lambda, m, z)$: It is identical to $\overline{H}_2^{\mathcal{M}}(\lambda, m, z)$ except that it uses v extracted from the left **WIPoK-Trap** as a witness in the left **WIPoK-2**.

Note that both $\overline{H}_2^{\mathcal{M}}(\lambda, m, z)$ and $\overline{H}_3^{\mathcal{M}}(\lambda, m, z)$ are efficient since they no longer need to output $\mathsf{val}_b(\widetilde{\tau})$. Then, by the assumption that the right **ExtCom-1-i^*** does not interleave with the left **WIPoK-2**, the WI property of the left **WIPoK-2** gives

$$\{\overline{H}_2^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*} \overset{c}{\approx} \{\overline{H}_3^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*}. \quad (73)$$

By the assumption that Bad 2 occurs and that $\mathcal{M}$ receives the right message of Step 1 at first, hybrids $H_2^{\mathcal{M}}(\lambda, m, z)$ and $H_3^{\mathcal{M}}(\lambda, m, z)$ do not rewind the right session at all. Indeed, they only rewind the left **WIPoK-Trap**, but the right message of Step 1 is sent before the left **WIPoK-Trap** starts by our simplifying assumption and the rest of messages in the right session is sent after the left **WIPoK-Trap** finishes by the assumption that Bad 2 occurs. Thus, by considering the combination of $\mathcal{M}$ and C as a single cheating committer, Lem. 14 gives

$$\{H_2^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*} \overset{c}{\approx} \{\overline{H}_2^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*} \quad (74)$$

[29] Rmk. 14 applies here.

and

$$\{H_3^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*} \overset{\mathrm{c}}{\approx} \{\overline{H}_3^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*}. \tag{75}$$

By combining Eq. (73) to (75), we complete the proof of Claim 16.

□

Claim 17. *It holds that*

$$\{H_3^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*} \overset{\mathrm{c}}{\approx} \{H_4^{\mathcal{M}}(\lambda, m, z)\}_{\lambda\in\mathbb{N}, m\in\{0,1\}^{\ell(\lambda)}, z\in\{0,1\}^*}.$$

Proof of Claim 17. This follows from a similar argument to that in the proof of Claim 16. However, we remark that we cannot prove it at once since the *total* round complexity of **ExtCom-{1,2,3}** is larger than n_5, in which case we cannot choose i^* such that the right **ExtCom-1-i^*** does not interleave with any message of the left **ExtCom-{1,2,3}**. To resolve this issue, we replace the committed message m with 0 for each execution of ExtCom in **ExtCom-{1,2,3}** one by one. Since we assume that n_5 is larger than the round complexity of ExtCom, a similar argument to the proof of Claim 16 works by using the computational indistinguishability of ExtCom.

□

Claim 18. *It holds that*

$$\{H_4^{\mathcal{M}}(\lambda, m_0, z)\}_{\lambda, m_0, m_1, z} \overset{\mathrm{c}}{\approx} \{H_4^{\mathcal{M}}(\lambda, m_1, z)\}_{\lambda, m_0, m_1, z},$$

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

Proof of Claim 18. This immediately follows from the computational hiding property of Naor's commitment noting that the left message m is only used for generating Naor's commitment in Step 2 in $H_4^{\mathcal{M}}(\lambda, m, z)$.

□

By combining Claims 15 to 18, we obtain

$$\{H_1^{\mathcal{M}}(\lambda, m_0, z)\}_{\lambda, m_0, m_1, z} \overset{\mathrm{c}}{\approx} \{H_1^{\mathcal{M}}(\lambda, m_1, z)\}_{\lambda, m_0, m_1, z}$$

where both ensembles are indexed by $\lambda \in \mathbb{N}$, $(m_0, m_1) \in \{0,1\}^{\ell(\lambda)} \times \{0,1\}^{\ell(\lambda)}$, and $z \in \{0,1\}^*$. This completes the proof of Lem. 16.

□

6.6.3 Analysis of Bad 3

Lemma 17. *The commitment scheme $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$ in Prot. 3 is non-malleable against asynchronous PPT adversaries whose schedule satisfies* Bad 3.

Proof. Recall that Bad 3 means that the right **WIPoK-1-A** starts before the left message of Step 4 is sent.

We consider the following sequence of hybrids.

– Hybrid $H_1^{\mathcal{M}}(\lambda, m, z)$: This hybrid is the real man-in-the-middle experiment $\mathsf{mim}^{\mathcal{M}}_{\langle C,R\rangle}(\lambda, m, z)$. The output of the experiment can be written as $(\mathsf{OUT}_{\mathcal{M}}, \mathsf{val}_b(\widetilde{\tau}))$ where $\mathsf{OUT}_{\mathcal{M}}$ is $\mathcal{M}$'s final output, b is the R's decision bit, and $\widetilde{\tau}$ is Naor's commitment generated in Steps 1 and 2 in the right session, and

$$\mathsf{val}_b(\widetilde{\tau}) := \begin{cases} \mathsf{val}(\widetilde{\tau}) & b = \top \\ \bot & b = \bot \end{cases}.$$

– Hybrid $H_2^{\mathcal{M}}(\lambda, m, z)$: Choose the smallest $i^* \in [n_5]$ such that the right **ExtCom-1-i^*** does not interleave with Steps 1 to 4 in the left, i.e., no message of the left **WIPoK-2** is sent during the execution of the right **ExtCom-1-i^***.[30] Such i^* exists by the pigeonhole principle since we assume that n_5 is larger than the total round complexity of Steps 1 to 4. This experiment is identical to $H_1^{\mathcal{M}}(\lambda, m, z)$ except that it runs the extractor of the right **ExtCom-1-i^*** to extract $\widetilde{m}^*$, and outputs $(\mathsf{OUT}_{\mathcal{M}}, \Gamma_b(\widetilde{m}^*))$ instead of $(\mathsf{OUT}_{\mathcal{M}}, \mathsf{val}_b(\widetilde{\tau}))$ where

$$\Gamma_b(m^*) := \begin{cases} m^* & b = \top \\ \bot & b = \bot \end{cases}.$$

Claim 19. *It holds that*

$$\{H_1^{\mathcal{M}}(\lambda, m, z)\}_{\lambda \in \mathbb{N}, m \in \{0,1\}^{\ell(\lambda)}, z \in \{0,1\}^*} \overset{s}{\approx} \{H_2^{\mathcal{M}}(\lambda, m, z)\}_{\lambda \in \mathbb{N}, m \in \{0,1\}^{\ell(\lambda)}, z \in \{0,1\}^*}.$$

Proof of Claim 19. This immediately follows from Lem. 14.

□

Claim 20. *It holds that*

$$\{H_2^{\mathcal{M}}(\lambda, m_0, z)\}_{\lambda, m_0, m_1, z} \overset{s}{\approx} \{H_2^{\mathcal{M}}(\lambda, m_1, z)\}_{\lambda, m_0, m_1, z}$$

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

Proof. By the choice of i^* and the assumption that Bad 3 occurs, $H_2^{\mathcal{M}}(\lambda, m, z)$ does not rewind the left session at all. Therefore, by considering the combination of $\mathcal{M}$ and R as a single cheating receiver, Claim 20 follows from the computational hiding property of Prot. 3.

□

By combining Claims 19 and 20, we obtain

$$\{H_1^{\mathcal{M}}(\lambda, m_0, z)\}_{\lambda, m_0, m_1, z} \overset{c}{\approx} \{H_1^{\mathcal{M}}(\lambda, m_1, z)\}_{\lambda, m_0, m_1, z}$$

where both ensembles are indexed by $\lambda \in \mathbb{N}$, $m_0, m_1 \in \{0,1\}^{\ell(\lambda)}$, and $z \in \{0,1\}^*$. This completes the proof of Lem. 17.

□

6.6.4 Analysis of Bad 4

Lemma 18. *The commitment scheme $\langle C, R \rangle_{\mathsf{tg}}^{\mathsf{async}}$ in Prot. 3 is non-malleable against asynchronous PPT adversaries whose schedule satisfies* Bad 4.

Proof. Recall that Bad 4 means that the right **WIPoK-1-B** starts before the left **WIPoK-1-A** finishes. Lem. 18 can be proven similarly to Lem. 17 except that we extract $\widetilde{m}$ from the right **ExtCom-2-i^*** that does not interleave with Steps 1 to 6 in the left.

□

[30] Rmk. 14 applies here.

6.6.5 Analysis of Bad 5

Lemma 19. *The commitment scheme $\langle C, R\rangle_{\mathsf{tg}}^{\mathsf{async}}$ in Prot. 3 is non-malleable against asynchronous PPT adversaries whose schedule satisfies* Bad 5.

Proof. Recall that Bad 5 means that the right **WIPoK-2** starts before the left **WIPoK-1-B** finishes. Lem. 19 can be proven similarly to Lem. 17 except that we extract $\widetilde{m}$ from the right **ExtCom-3-**i^* that does not interleave with Steps 1 to 10 in the left. □

By combining Lem. 12 and 15 to 19 along with Lem. 11, we complete the proof of Thm. 13.

7 Extract-and-Simulate Lemma

In this section, we show a generalized version of the extract-and-simulate lemma given in [CCLY22, Lemma 4].

Lemma 20 (Extract-and-Simulate 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 $b \in \{\top, \bot\}$ 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, an error parameter $1^{\gamma^{-1}}$, 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, γ, ρ_λ, and z_λ, the output s of $\mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, \rho_\lambda, z_\lambda)$ is equal to $s^*_{z_\lambda}$ whenever $s \neq \bot$.*
2. *For any noticeable function $\gamma(\lambda)$, there exists a noticeable function $\delta(\lambda)$ that is efficiently computable from $\gamma(\lambda)$ and satisfies the following. For any sequence $\{\rho_\lambda, z_\lambda\}_{\lambda\in\mathbb{N}}$ of polynomial-size quantum states and classical strings, if*

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

then

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

Then, there exists a QPT algorithm $\mathcal{SE}$ and an efficiently computable polynomial poly *such that for any sequence $\{\rho_\lambda, z_\lambda\}_{\lambda\in\mathbb{N}}$ of polynomial-size quantum states and classical strings and noticeable function $\varepsilon = \varepsilon(\lambda)$,*

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

where $\gamma = \mathsf{poly}(\varepsilon)$ and $\Gamma_b(s^*_{z_\lambda}) := \begin{cases} s^*_{z_\lambda} & \text{if } b = \top \\ \bot & \text{otherwise} \end{cases}$.

In [CCLY22], the authors essentially proved a special case of the above lemma where the simulation-less extractor is fixed to Unruh's rewinding extractor [Unr12]. We observe that essentially the same proof works for general simulation-less extractors.[31] We provide the full proof of Lem. 20 in Appx. A for completeness.

[31] Indeed, this observation is also mentioned in the technical overview of [CCLY22].

8 Post-Quantum Non-Malleable Commitments

In this section, we construct a full-fledged post-quantum non-malleable commitment scheme that supports an exponential-size tag space and is secure against QPT asynchronous MIM adversaries.

8.1 Small-Tag, One-sided, Synchronous, Post-Quantum Setting

First, we show that the Prot. 1 given in Sec. 4 is post-quantumly secure if we instantiate it with post-quantum building-blocks. Specifically, let $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$ be Prot. 1 instantiated with post-quantum Naor's commitment Com (which is constructed based on post-quantum OWFs), post-quantum injective OWF f, and a post-quantum witness-distinguishable arguments of knowledge with ε-close emulation as per Def. 14 (see Prot. 5 in Appx. B for the full description of $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$). Then, we can prove the following theorem:

Theorem 21. *The commitment scheme $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$ is non-malleable against one-sided synchronous QPT adversaries with tag space $[n]$, with n being any polynomial on λ.*

We remark that the injectivity assumption for f can be removed in exactly the same way as in the classical setting (as explained in Sec. 4.8).

The proof of Thm. 21 is very similar to that of its classical counterpart (Thm. 2) except for one step as explained below. Thus, we only give a proof sketch below, and defer the full proof to Appx. B.

Proof of Thm. 21 (sketch). We observe that all steps of the proof of Thm. 2, except for the proof of Lem. 7 (in particular, Lem. 9 in Sec. 4.4), can be translated into the post-quantum setting easily with the following remarks. The only difference in those steps (except for some superficial differences like PPT vs QPT) is that we only have WIAoK with ε-close emulation (as per Def. 14) instead of WIPoK (with negligibly-close emulation). First, the AoK property instead of the PoK property suffices because we run witness-extended emulators only for (possibly non-uniform) efficient cheating provers. Second, the ε-close emulation (in contrast to negligibly-close emulation) also suffices, because we can take a sufficiently small noticeable error parameter such that it is much smaller than the MIM adversary's advantage. With these modifications, the whole proof will go through.

Thus, the only remaining issue is how to prove the post-quantum version of Lem. 7 (using a post-quantum version of Lem. 9). Roughly, we have to construct a simulation-extractor $\mathcal{SE}$ that extracts the right committed message $\mathsf{val}(\widetilde{\tau})$ while simulating $\mathcal{M}$'s final state from a simulation-less extractor $\mathcal{K}$, which extracts $\mathsf{val}(\widetilde{\tau})$ with a noticeable probability *without simulating $\mathcal{M}$'s final state*. This task is exactly what can be achieved by our new extract-and-simulate lemma (Lem. 20). Indeed, the requirements for $\mathcal{K}$ in Lem. 9 (more precisely, its post-quantum version Lem. 30) exactly correspond to the assumptions of Lem. 20. Therefore, this step can be completed by reducing it to Lem. 20. (We believe that readers with good understanding of the classical case can directly go to *Proof of Lem. 28* in Page 84 after checking the statements of Lem. 28 and 30 and relevant definitions to see how this step exactly works.)

□

8.2 Small-Tag, Asynchronous, Post-Quantum Setting

Next, we explain how to make $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$ (shown in Prot. 5) secure against two-sided asynchronous adversaries. We emphasize that this subsection still focuses on the small-tag setting. This step is essentially the same as its classical counterpart in Sec. 5 and 6 except for that we rely on post-quantum building blocks.

We present our construction in Prot. 4. This protocol makes use of the following building blocks.

- A post-quantum OWF f;
- Naor's commitment Com that is implemented with a post-quantum OWF;
- A post-quantum witness-indistinguishable argument of knowledge with ε-close emulation WIAoK (as per Def. 14);
- A post-quantum ε-simulatable extractable commitment ExtCom (as per Def. 9). We denote by $\mathsf{Verify}_{\mathsf{ExtCom}}$ the verification algorithm of ExtCom in the decommit stage.

Similarly to Sec. 6, we recursively define $n_5, ..., n_9$ so that n_i is greater than the total round complexity of Steps 1 to $i-1$. We also require that n_5 is greater than the round complexities of WIAoK and ExtCom. It is easy to see that we can set n_i to a constant if both WIPoK and ExtCom have constant rounds. In particular, Prot. 4 runs in constant rounds.

Protocol 4: Post-Quantum Small-Tag Asynchronous NMCom $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg,PQ}}$

The tag space is defined to be $[n]$ where n is a polynomial on λ. Let $t \in [n]$ be the tag for the following interaction. Let m be the message to be committed to.

Commit Stage:

1. Receiver R samples and sends the first message β for Naor's commitment.
2. Committer C commits to m using the second message of Naor's commitment. C also sends a β' that will be used as the first Naor's message for R to generate a commitment in next step. Formally, C samples r and β' and sends the tuple $\big(\mathsf{com} = \mathsf{Com}_{\beta}(m; r),\ \beta'\big)$.
3. R and C do the following:
   (a) **(TrapGen.)** R computes $V_0 = f(v_0)$ and $V_1 = f(v_1)$ with $v_0, v_1 \xleftarrow{\$} \{0,1\}^\lambda$ and sends (V_0, V_1) to C
   (b) **(WIAoK-Trap.)** R and C execute an instance of WIAoK where R proves to C that he "knows" a pre-image of V_0 or V_1. Formally, R proves that $(V_0, V_1) \in \mathcal{L}^{\mathrm{OR}}_f$, where
   $$\mathcal{L}^{\mathrm{OR}}_f \coloneqq \big\{(V_0, V_1) \bigm| \exists v\ s.t.\ \ f(v) = V_0\ \vee\ f(v) = V_1\big\}. \tag{76}$$
   Note that R uses v_0 as the witness when executing this WIAoK.
4. R performs the following computation:
   (a) R computes $\{y^A_i = f(x^A_i)\}_{i\in[t]}$ with $x^A_i \xleftarrow{\$} \{0,1\}^\lambda$ for each $i \in [t]$, and sets $Y^A \coloneqq (y^A_1, \ldots, y^A_t)$. R also computes $\{\mathsf{com}^A_i = \mathsf{Com}_{\beta'}(x^A_i; r^A_i)\}_{i\in[t]}$, where r^A_i is a random string for each $i \in [t]$.
   (b) R computes $\{y^B_i = f(x^B_i)\}_{i\in[n-t]}$ with $x^B_i \xleftarrow{\$} \{0,1\}^\lambda$ for each $i \in [n-t]$, and sets $Y^B \coloneqq (y^B_1, \ldots, y^B_{n-t})$. R also computes $\{\mathsf{com}^B_i = \mathsf{Com}_{\beta'}(x^B_i; r^B_i)\}_{i\in[n-t]}$, where r^B_i is a random string for each $i \in [n-t]$.

   R sends the tuple $\big(Y^A,\ \{\mathsf{com}^A_i = \mathsf{Com}_{\beta'}(x^A_i; r^A_i)\}_{i\in[t]},\ Y^B,\ \{\mathsf{com}^B_i = \mathsf{Com}_{\beta'}(x^B_i; r^B_i)\}_{i\in[n-t]}\big)$.
5. **(ExtCom-1 $\times$ n_5.)** C and R sequentially execute n_5 instances of ExtCom where C commits to m. Let τ^1_i and decom^1_i be the transcript and decommitment (privately obtained by C) of the i-th execution.
6. **(WIAoK-1-A $\times$ n_6.)** R and C sequentially execute n_6 instances of WIAoK where R proves to C that he "knows" a pre-image of some y^A_i contained in Y^A, and y^A_i are consistent with com^A_i

(as defined in Step 4a). Formally, R proves that $(Y^A, \{\mathsf{com}_i^A\}_{i\in[t]}) \in \mathcal{L}^t_{f,\beta'}$, where

$$\mathcal{L}^t_{f,\beta'} := \left\{ (y_1^A,\ldots,y_t^A), (\mathsf{com}_1^A,\ldots,\mathsf{com}_t^A) \;\middle|\; \exists (i, x_i^A, r_i^A) \ s.t. \begin{array}{l} i \in [t] \ \wedge \\ y_i^A = f(x_i^A) \ \wedge \\ \mathsf{com}_i^A = \mathsf{Com}_{\beta'}(x_i^A; r_i^A) \end{array} \right\}. \tag{77}$$

Note that R uses $(1, x_1^A, r_1^A)$ as the witness when executing this WIAoK.

7. **(ExtCom-2 $\times$ n_7.)** C and R sequentially execute n_7 instances of ExtCom where C commits to m. Let τ_i^2 and decom_i^2 be the transcript and decommitment (privately obtained by C) of the i-th execution.

8. **(WIAoK-1-B $\times$ n_8)** R and C sequentially execute n_8 instances of WIAoK where R proves to C that he "knows" a pre-image of some y_i^B contained in Y^B, and y_i^B are consistent with com_i^B (as defined in Step 4b). Formally, R proves that $(Y^B, \{\mathsf{com}_i^B\}_{i\in[n-t]}) \in \mathcal{L}^{n-t}_{f,\beta'}$, where

$$\mathcal{L}^{n-t}_{f,\beta'} := \left\{ (y_1^B,\ldots,y_{n-t}^B), (\mathsf{com}_1^B,\ldots,\mathsf{com}_{n-t}^B) \;\middle|\; \exists (i, x_i^B, r_i^B) \ s.t. \begin{array}{l} i \in [n-t] \ \wedge \\ y_i^B = f(x_i^B) \ \wedge \\ \mathsf{com}_i^B = \mathsf{Com}_{\beta'}(x_i^B; r_i^B) \end{array} \right\}. \tag{78}$$

Note that R uses $(1, x_1^B, r_1^B)$ as the witness when executing this WIPoK.

9. **(ExtCom-3 $\times$ n_9.)** C and R sequentially execute n_9 instances of ExtCom where C commits to m. Let τ_i^2 and decom_i^2 be the transcript and decommitment (privately obtained by C) of the i-th execution.

10. **(WIAoK-2.)** C and R execute an instance of WIAoK where C proves to R that he "knows" the (same) message committed in com (defined in Step 2) and all commitments τ_i^c of ExtCom (in Steps 5, 7 and 9), *or* a pre-image of some y_i^A contained in Y^A (defined in Step 4a), *or* a pre-image of some y_i^B contained in Y^B (defined in Step 4b) *or* a preimage of either of V_0 or V_1 (defined in Step 3a). Formally, C proves that $(\mathsf{com}, \{\tau_i^c\}_{(c,i)\in J}, Y^A, Y^B, V_0, V_1) \in \mathcal{L}_\beta \vee \mathcal{L}_f^t \vee \mathcal{L}_f^{n-t} \vee \mathcal{L}_f^{\mathrm{OR}}$, where $J = (\{1\}\times[n_5]) \cup (\{2\}\times[n_7]) \cup (\{3\}\times[n_9])$,

$$\mathcal{L}_\beta := \left\{ (\mathsf{com}, \{\tau_i^c\}_{(c,i)\in J}) \;\middle|\; \begin{array}{l} \exists (m, r, \{\mathsf{decom}_i^c\}_{(c,i)\in J}) \ s.t. \\ \mathsf{com} = \mathsf{Com}_\beta(m; r) \ \wedge \\ \forall\ (c,i) \in J,\ \mathsf{Verify}_{\mathsf{ExtCom}}(m, \tau_i^c, \mathsf{decom}_i^c) = \top \end{array} \right\} \tag{79}$$

$$\mathcal{L}_f^t := \{(y_1^A,\ldots,y_t^A) \mid \exists (i, x_i^A) \ s.t. \ i\in[t] \wedge y_i^A = f(x_i^A)\}, \tag{80}$$

$$\mathcal{L}_f^{n-t} := \{(y_1^B,\ldots,y_{n-t}^B) \mid \exists (i, x_i^B) \ s.t. \ i\in[n-t] \wedge y_i^B = f(x_i^B)\}, \tag{81}$$

and $\mathcal{L}_f^{\mathrm{OR}}$ is defined in Language (76). Note that C uses the $(m, r, \{\mathsf{decom}_i^c\}_{(c,i)\in J})$ defined in Steps 2, 5, 7 and 9 as the witness when executing this WIAoK.

Decommit Stage: C sends (m, r). R accepts if $\mathsf{com} = \mathsf{Com}_\beta(m; r)$, and rejects otherwise.

We remark that Prot. 4 is identical to Prot. 3 except that we require post-quantum security for building blocks and use WIAoK instead of WIPoK.

Security. Completeness is straightforward. Statistical binding follows from that of Naor's commitment.

For computational hiding and non-malleability, we observe that the proofs against classical adversaries in Sec. 6 is already quantum-friendly, and they can be translated into proofs against quantum adversaries in a straightforward manner. The only difference is that we now have ε-close simulation for WIAoK and ExtCom though the classical counterparts have negligible simulation errors. Thus, whenever we invoke extractors of WIPoK or ExtCom in the original proof in the classical setting, a noticeable error occurs in the post-quantum setting. However, this is not a problem since the error can be an *arbitrarily small* noticeable function. Thus, by repeating the same proofs as those in the classical case, we can show that QPT MIM adversary's advantage is smaller than *any* noticeable function in the security parameter. This means that the advantage is negligible, which is nothing but non-malleability in the standard sense. Thus, we obtain the following theorems.

Theorem 22. *The commitment scheme $\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$ in Prot. 4 is computationally hiding against QPT adversaries.*

Theorem 23. *The commitment scheme $\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$ in Prot. 4 is non-malleable against asynchronous QPT adversaries with tag space $[n]$, with n being any polynomial on λ.*

8.3 Tag Amplification

Finally, we explain how to amplify the tag space of Prot. 4. For this step, we rely on the following variant of a theorem shown in [BLS22].

Theorem 24 ([BLS22, Section 6]). *Assume the existence of*

- *a post-quantum, k_1-round, ε-simulatable extractable commitment (as per Def. 9), and*
- *a post-quantum, k_2-round, witness-indistinguishable argument.*

Let $\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$ be a post-quantumly non-malleable commitment scheme, where the subscript tg denotes the length *of the maximum tag it can support. Assume $\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$ satisfies the following conditions:*

1. *$\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$ supports $\mathsf{tg} \in \{3,4,\ldots,O(\log\lambda)\}$ bit tags, and*
2. *$\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$ is also a post-quantum (k_1+k_2)-robust ε-simulatable extractable commitment scheme (as per Def. 10).*

Then, there exists a post-quantum non-malleable commitment scheme $\langle C,R\rangle^{\mathsf{async}}_{\mathsf{TG},\mathsf{PQ}}$ which supports tags of $\mathsf{TG} = 2^{\mathsf{tg}-1}$ bits. Moreover, $\langle C,R\rangle^{\mathsf{async}}_{\mathsf{TG},\mathsf{PQ}}$ has $k_1+k_2+O(1)$ rounds.

Remark 16. In the original version of the above theorem in [BLS22], they require negl-simulatability instead of ε-simulatability for the extractable commitment and the non-malleable commitment schemes (in Condition 2). However, we can see that the ε-simulatable version suffices due to a similar reason as explained in the end of the previous subsection. That is, if we use extractors with a noticeable error in their proof, we can show that the adversary's advantage is at most a noticeable function in λ. Since the noticeable error can be chosen to be arbitrarily small, this implies non-malleability in the standard sense.

Given the above Thm. 24, we only need to prove that $\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$ in Prot. 4 has (k_1+k_2)-robust ε-simulatable extractability where k_1 and k_2 are the round complexities of ExtCom and WIAoK used in the construction of $\langle C,R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$.[32] We show this below.

[32] For applying Thm. 24, we only need to require computational soundness for the witness indistinguishable argument rather than the witness-extended emulation property. But since the latter is stronger than the former, we can simply instantiate it with any WIAoK that satisfies Def. 14.

Theorem 25. *The commitment scheme* $\langle C, R\rangle^{\mathsf{async}}_{\mathsf{tg},\mathsf{PQ}}$ *in Prot. 4 satisfies post-quantum* (k_1+k_2)*-robust* ε*-simulatable extractability (as per Def. 10) where* k_1 *and* k_2 *are round complexities of* ExtCom *and* WIAoK*.*

Proof. (sketch.) We first remark that we have $n_9 > k_1 + k_2$ since we assume that n_9 is larger than the total round complexity of Steps 1 to 8, which is clearly larger than $k_1 + k_2$ since those steps include repetitions of ExtCom and WIAoK. Then, by the pigeonhole principle, for each session with O^∞, there is i^* such that **ExtCom-3-**i^* does not interleave with the interaction with the external (k_1+k_2)-round machine B in the definition of r-robust ε-simulatable extractability (in Def. 10). Then, we can extract the committed message by running the extractor for **ExtCom-3-**i^* without rewinding B.[33] We can show that the extracted message is the correct committed message in com (in Step 1) whenever the receiver accepts except for a negligible probability similarly to Lem. 14. This enables us to simulate the oracle O^∞ without rewinding B.

$\square$

Finally, by combining Thm. 23 to 25, we obtain the following theorem.

Theorem 26. *Assuming the existence of post-quantum OWFs, there exist constant-round post-quantum non-malleable commitments supporting tag space* $[\Omega(2^\lambda)]$.

9 Application: Quantum-Secure MPC in Constant Rounds from QLWE

MPC for Classical Functionalities. Agarwal et al. [ABG+21] constructed a constant-round post-quantum MPC protocol in the plain model assuming the super-polynomial hardness of QLWE and a QLWE-based circular security assumption. The only reason why they rely on the super-polynomial hardness of QLWE is for their construction of a constant-round post-quantum non-malleable commitment scheme.[34] Since we have constructed constant-round post-quantum non-malleable commitments from post-quantum OWFs (Thm. 26), we can weaken the assumption to the polynomial hardness of QLWE. Thus, we obtain the following theorem.

Theorem 27. *Assuming the polynomial hardness of QLWE and a QLWE-based circular security assumption (as in [ABG+21]), there exist constant-round constructions of post-quantum MPC for classical functionalities in the plain model.*

Recall that "post-quantum MPC" means an MPC protocol secure against QPT adversaries where honest parties only need to perform classical computation. This is the first construction of constant-round post-quantum MPC from polynomial hardness assumptions in the plain model.

Remark 17 (Synchronous Security Suffices.). For the construction of post-quantum MPC, we only need post-quantum non-malleable commitments secure against *synchronous* adversaries. It is simpler to obtain such non-malleable commitments than those against asynchronous adversaries. For example, this can be done by applying the tag amplification of [ABG+21, Sec. 7.3] to our small-tag, synchronous construction (i.e., Prot. 2 instantiated with post-quantum building blocks).

[33] We can assume that such i^* is known in advance w.l.o.g. by Lem. 11.

[34] Actually, what they need is the so-called *many-to-one* non-malleable commitments. But it is known that (one-to-one) non-malleability as defined in Def. 7 is equivalent to many-to-one non-malleability (even in the post-quantum setting, as noted in [ABG+21, Lemma 7.3]).

MPC for Quantum Functionalities. Recently, Bartusek et al. [BCKM21a] built a constant-round quantum-secure MPC for quantum functionalities *in the CRS model*[35], based on the hardness of QLWE. It is easy to see that the post-quantum MPC from Thm. 27 can be used to instantiate the CRS required by the [BCKM21a] protocol. Since the protocol from Thm. 27 is also constant round, this leads to the first constant-round quantum-secure MPC for quantum functionalities from polynomial hardness assumptions *without any trusted setup.*

Theorem 28. *Assuming (polynomial) QLWE and the QLWE-based circular security assumption (as in [ABG$^+$21]), there exists a constant-round construction of quantum-secure MPC for quantum functionalities in the plain model.*

References

Aar05. Scott Aaronson. Limitations of quantum advice and one-way communication. *Theory of Computing*, 1(1):1–28, 2005. 16

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, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 435–464. Springer, Heidelberg, October 2021. 2, 4, 68, 69

Bar01. Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd FOCS*, pages 106–115. IEEE Computer Society Press, October 2001. 1

Bar02. Boaz Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In *43rd FOCS*, pages 345–355. IEEE Computer Society Press, November 2002. 1

BCKM21a. James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. On the round complexity of secure quantum computation. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 406–435, Virtual Event, August 2021. Springer, Heidelberg. 4, 69

BCKM21b. James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. One-way functions imply secure computation in a quantum world. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 467–496, Virtual Event, August 2021. Springer, Heidelberg. 2

BDH$^+$17. Brandon Broadnax, Nico Döttling, Gunnar Hartung, Jörn Müller-Quade, and Matthias Nagel. Concurrently composable security with shielded super-polynomial simulators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 351–381. Springer, Heidelberg, April / May 2017. 1

BGJ$^+$18. Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Promise zero knowledge and its applications to round optimal MPC. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 459–487. Springer, Heidelberg, August 2018. 1

BL18. Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 209–234. Springer, Heidelberg, November 2018. 1

BLS22. Nir Bitansky, Huijia Lin, and Omri Shmueli. Non-malleable commitments against quantum attacks. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 519–550. Springer, 2022. 2, 3, 5, 17, 18, 21, 22, 54, 67

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 ACM STOC*, pages 269–279. ACM Press, June 2020. 2, 72

BZ13. Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 361–379. Springer, Heidelberg, August 2013. 2

Can00. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. 1

[35] More accurately, [BCKM21a] presented their construction in the OT-hybrid model. But it is known that the OT functionality can be instantiated using the post-quantum straight-line simulatable construction in the CRS model from [PVW08], which is also constant-round and based on the hardness of QLWE.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. 1

CCG$^+$19. Arka Rai Choudhuri, Michele Ciampi, Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Round optimal secure multiparty computation from minimal assumptions. Cryptology ePrint Archive, Report 2019/216, 2019. `https://eprint.iacr.org/2019/216`. 1

CCLY22. Nai-Hui Chia, Kai-Min Chung, Xiao Liang, and Takashi Yamakawa. Post-quantum simulatable extraction with minimal assumptions: Black-box and constant-round. In *Annual International Cryptology Conference*. Springer, 2022. 2, 3, 13, 14, 15, 16, 19, 21, 22, 24, 63, 72, 73

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, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 315–345, Virtual Event, August 2021. Springer, Heidelberg. 16, 76

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002. 1

CLP20. Rohit Chatterjee, Xiao Liang, and Omkant Pandey. Improved black-box constructions of composable secure computation. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *ICALP 2020*, volume 168 of *LIPIcs*, pages 28:1–28:20. Schloss Dagstuhl, July 2020. 1

COSV17. Michele Ciampi, Rafail Ostrovsky, Luisa Siniscalchi, and Ivan Visconti. Four-round concurrent non-malleable commitments from one-way functions. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 127–157. Springer, Heidelberg, August 2017. 1, 3

DDN91. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991. 1, 3

FP96. Christopher A Fuchs and Asher Peres. Quantum-state disturbance versus information gain: Uncertainty relations for quantum information. *Physical Review A*, 53(4):2038, 1996. 1

GLOV12. Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd FOCS*, pages 51–60. IEEE Computer Society Press, October 2012. 1, 3

GMPP16. Sanjam Garg, Pratyay Mukherjee, Omkant Pandey, and Antigoni Polychroniadou. The exact round complexity of secure computation. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 448–476. Springer, Heidelberg, May 2016. 1

Gol01. Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. 22, 23

Goy11. Vipul Goyal. Constant round non-malleable protocols using one way functions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 695–704. ACM Press, June 2011. 1, 3

GPR16. Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1128–1141. ACM Press, June 2016. 1, 3, 20

GR19. Vipul Goyal and Silas Richelson. Non-malleable commitments using Goldreich-Levin list decoding. In David Zuckerman, editor, *60th FOCS*, pages 686–699. IEEE Computer Society Press, November 2019. 1, 3

GRRV14. Vipul Goyal, Silas Richelson, Alon Rosen, and Margarita Vald. An algebraic approach to non-malleability. In *55th FOCS*, pages 41–50. IEEE Computer Society Press, October 2014. 1, 3

HSS11. Sean Hallgren, Adam Smith, and Fang Song. Classical cryptographic protocols in a quantum world. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 411–428. Springer, Heidelberg, August 2011. 14

Khu17. Dakshita Khurana. Round optimal concurrent non-malleability from polynomial hardness. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part II*, volume 10678 of *LNCS*, pages 139–171. Springer, Heidelberg, November 2017. 1

Khu21. Dakshita Khurana. Non-interactive distributional indistinguishability (NIDI) and non-malleable commitments. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 186–215. Springer, Heidelberg, October 2021. 1

KOS03. Jonathan Katz, Rafail Ostrovsky, and Adam Smith. Round efficiency of multi-party computation with a dishonest majority. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 578–595. Springer, Heidelberg, May 2003. 1

KS17. Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In Chris Umans, editor, *58th FOCS*, pages 564–575. IEEE Computer Society Press, October 2017. 1, 4

Lin03. Yehuda Lindell. Parallel coin-tossing and constant-round secure two-party computation. *Journal of Cryptology*, 16(3):143–184, June 2003. 7, 22, 23

LMS21. Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited (or: How to do quantum rewinding undetectably). Cryptology ePrint Archive, Report 2021/1543, 2021. `https://ia.cr/2021/1543`. 3, 14, 21, 24

LN11. Carolin Lunemann and Jesper Buus Nielsen. Fully simulatable quantum-secure coin-flipping and applications. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 21–40. Springer, Heidelberg, July 2011. 14

LP09. Huijia Lin and Rafael Pass. Non-malleability amplification. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 189–198. ACM Press, May / June 2009. 1

LP11. Huijia Lin and Rafael Pass. Constant-round non-malleable commitments from any one-way function. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 705–714. ACM Press, June 2011. 1, 3

LPS17. Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Chris Umans, editor, *58th FOCS*, pages 576–587. IEEE Computer Society Press, October 2017. 1

LPV08. Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 571–588. Springer, Heidelberg, March 2008. 1, 20

MPR06. Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th FOCS*, pages 367–378. IEEE Computer Society Press, October 2006. 1

Pas03. Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2003. 1

Pas13. Rafael Pass. Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 334–354. Springer, Heidelberg, March 2013. 1

PPV08. Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 57–74. Springer, Heidelberg, August 2008. 1

PR05. Rafael Pass and Alon Rosen. New and improved constructions of non-malleable cryptographic protocols. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 533–542. ACM Press, May 2005. 1, 46

PS04. Manoj Prabhakaran and Amit Sahai. New notions of security: Achieving universal composability without trusted setup. In László Babai, editor, *36th ACM STOC*, pages 242–251. ACM Press, June 2004. 1

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008. 69

PW09. Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418. Springer, Heidelberg, March 2009. 13, 21

PW10. Rafael Pass and Hoeteck Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 638–655. Springer, Heidelberg, May / June 2010. 4

Unr12. Dominique Unruh. Quantum proofs of knowledge. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 135–152. Springer, Heidelberg, April 2012. 63

Wat06. John Watrous. Zero-knowledge against quantum attacks. In Jon M. Kleinberg, editor, *38th ACM STOC*, pages 296–305. ACM Press, May 2006. 2

Wat09. John Watrous. Zero-knowledge against quantum attacks. *SIAM Journal on Computing*, 39(1):25–58, 2009. 72

Wee10. Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. In *51st FOCS*, pages 531–540. IEEE Computer Society Press, October 2010. 1, 5

WZ82. William K Wootters and Wojciech H Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982. 1

Zha12. Mark Zhandry. How to construct quantum random functions. In *53rd FOCS*, pages 679–687. IEEE Computer Society Press, October 2012. 2

Appendix

A Proof of Extract-and-Simulate Lemma (Lem. 20)

We prove Lem. 20. The proof follows similar techniques as in the proof of [CCLY22, Lemma 4].

A.1 Preparation

We prepare several lemmas that will be used in the proof of Lem. 20.

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

Lemma 21 (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 *(in a register outside* Inp*) 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_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* $p_0, q \in (0,1)$*,* $\alpha \in (0, \frac{1}{2})$ *such that:*

- *Amplification executes for enough time:* $T \geq \frac{\log(1/\alpha)}{4p_0(1-p_0)}$*,*
- *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* α *distance: For every* n*-qubit state* ρ*,* $|p(\rho) - q| < \alpha$*, and*
- q *is closer to* $\frac{1}{2}$*:* $p_0(1-p_0) \leq q(1-q)$*,*

then for every n*-qubit state* ρ*,*

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

Lemma 22 ([CCLY22, Lemma 7]). *Let* $\mathcal{A}_{\mathsf{CCY}}$ *be a QPT algorithm (possibly with classical advice) that takes a quantum input* $|\psi\rangle \in \mathcal{H}$ *and outputs a classical string* s *or* $\perp$*. Suppose that there exists* s^* *such that for any* $|\psi\rangle \in \mathcal{H}$*,* $\mathcal{A}_{\mathsf{CCY}}(|\psi\rangle)$ *outputs* s^* *whenever* $\mathcal{A}_{\mathsf{CCY}}$ *does not output* $\perp$*. For any noticeable* δ*, there exists an orthogonal decomposition of* $\mathcal{H}$ *into* $S_{<\delta}$ *and* $S_{\geq\delta}$ *that satisfies the following.*

1. *For any normalized state* $|\psi_{<\delta}\rangle \in S_{<\delta}$*, it holds that* $\Pr[\mathcal{A}_{\mathsf{CCY}}(|\psi_{<\delta}\rangle) = s^*] < \delta$*. Similarly, for any normalized state* $|\psi_{\geq\delta}\rangle \in S_{\geq\delta}$*, it holds that* $\Pr[\mathcal{A}_{\mathsf{CCY}}(|\psi_{\geq\delta}\rangle) = s^*] \geq \delta$*.*
2. *There exists a QPT algorithm* $\mathsf{Ext}_{\mathsf{CCY}}$ *(whose description does not depend on that of* $\mathcal{A}_{\mathsf{CCY}}$*) that satisfies the following.*

(a) For any normalized state $\left|\psi_{\geq\delta}\right\rangle \in S_{\geq\delta}$*, it holds that*

$$\Pr\left[\begin{array}{l} b_{\mathsf{CCY}} = \top \ \wedge \ s = s^* \ \wedge \\ \| \left|\psi_{\geq\delta}\right\rangle \left\langle\psi_{\geq\delta}\right| - \left|\psi'_{\geq\delta}\right\rangle \left\langle\psi'_{\geq\delta}\right| \|_{tr} = \mathsf{negl}(\lambda) \end{array} : (b_{\mathsf{CCY}}, s, \left|\psi'_{\geq\delta}\right\rangle) \leftarrow \mathsf{Ext}_{\mathsf{CCY}}(1^{\delta^{-1}}, \mathcal{A}_{\mathsf{CCY}}, \left|\psi_{\geq\delta}\right\rangle)\right]$$
$$= 1 - \mathsf{negl}(\lambda)$$

where the probability is over intermediate measurement results by $\mathsf{Ext}_{\mathsf{CCY}}$.[36]

(b) For any normalized state $|\psi_{<\delta}\rangle \in S_{<\delta}$*,*

$$\Pr\Big[b_{\mathsf{CCY}} = \bot \ \vee \ \left|\psi'_{<\delta}\right\rangle \in S_{<\delta} : (b_{\mathsf{CCY}}, s, \left|\psi'_{<\delta}\right\rangle) \leftarrow \mathsf{Ext}_{\mathsf{CCY}}(1^{\delta^{-1}}, \mathcal{A}_{\mathsf{CCY}}, |\psi_{<\delta}\rangle)\Big] = 1$$

where the probability is over intermediate measurement results by $\mathsf{Ext}_{\mathsf{CCY}}$.

Lemma 23 ([CCLY22, Lemma 8]). *Let* $|\phi_b\rangle = \left|\phi_{b,0}\right\rangle + \left|\phi_{b,1}\right\rangle$ *be a quantum state in a Hilbert space* $\mathcal{H}$ *such that* $\left\langle\phi_{b,0}\middle|\phi_{b,1}\right\rangle = 0$ *for* $b \in \{0,1\}$*. Let* F *be a quantum algorithm that takes a state in* $\mathcal{H}$ *as input and outputs a quantum state (not necessarily in* $\mathcal{H}$*) or a classical failure symbol* Fail*. Suppose that we have*

$$\Pr\left[F\left(\frac{\left|\phi_{b,0}\right\rangle \left\langle\phi_{b,0}\right|}{\| \left|\phi_{b,0}\right\rangle \|^2}\right) = \mathsf{Fail}\right] \geq 1 - \gamma$$

for $b \in \{0,1\}$ *where* $\gamma < 1/36$*. Then for any distinguisher* D*, it holds that*

$$|\Pr[D(F(|\phi_0\rangle \langle\phi_0|)) = 1] - \Pr[D(F(|\phi_1\rangle \langle\phi_1|)) = 1]| \leq 4\gamma^{1/4} + \| |\phi_{1,1}\rangle \langle\phi_{1,1}| - |\phi_{0,1}\rangle \langle\phi_{0,1}| \|_{tr}.$$

A.2 Proving Lem. 20

Since any mixed state can be seen as a distribution over pure states, we assume $\mathcal{G}$'s input ρ_λ is a pure state and denote it by $|\psi\rangle$, omitting the dependence on λ. Similarly, we simply write z to mean z_λ and s^* to mean $s^*_{z_\lambda}$ for simplicity.

For a noticeable function $\gamma(\lambda)$ and a quantum state $|\psi\rangle$, we define an experiment $\mathsf{Exp}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$ as follows

$\mathsf{Exp}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$: Run $(b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$.

- If $b = \top$, the experiment outputs $(\rho_{\mathsf{out}}, s^*)$. We remark that this step may not be done efficiently since we do not assume that s^* can be computed from z efficiently.
- If $b = \bot$, the experiment outputs $(\rho_{\mathsf{out}}, \bot)$.

What we have to do is to construct a QPT $\mathcal{SE}$ such that for any polynomial-size $|\psi\rangle$ and noticeable $\varepsilon = \varepsilon(\lambda)$,

$$\{\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)\}_{\lambda\in\mathbb{N}} \overset{\mathrm{s}}{\approx}_\varepsilon \{(\mathsf{Exp}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)\}_{\lambda\in\mathbb{N}}$$

for some $\gamma = \mathsf{poly}(\varepsilon)$.

Let $\mathsf{Exp}_\bot(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$ and $\mathsf{Exp}_\top(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$ be the same as $\mathsf{Exp}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$ except that they output a failure symbol Fail in the cases of $b = \top$ and $b = \bot$, respectively. That is, they work as follows.

$\mathsf{Exp}_\bot(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$: Run $(b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$.

[36] We sometimes consider an output of a quantum algorithm as a mixed state taking all the randomness into account. On the other hand, here we consider the output of $\mathsf{Ext}_{\mathsf{CCY}}$ as a probabilistic variable that takes pure states. This implicitly assumes that the third output of $\mathsf{Ext}_{\mathsf{CCY}}$ is a pure state whenever the third input is a pure state.

– If $b = \top$, the experiment outputs Fail.

– If $b = \bot$, the experiment outputs $(\rho_{\mathsf{out}}, \bot)$.

$\mathsf{Exp}_{\top}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$: Run $(b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^{\lambda}, 1^{\gamma^{-1}}, |\psi\rangle, z)$.

– If $b = \top$, the experiment outputs $(\rho_{\mathsf{out}}, s^*)$.

– If $b = \bot$, the experiment outputs Fail.

First, we give simulation extractors for each of these experiments.

Lemma 24 (Extract-and-Simulate for the Case of $b = \bot$). *For any efficiently computable $\gamma = \mathsf{poly}(\varepsilon)$, there is a QPT algorithm $\mathcal{SE}_{\bot}$ such that for any polynomial-size quantum state $|\psi\rangle$ and noticeable ε,*

$$\{\mathcal{SE}_{\bot}(1^{\lambda}, 1^{\varepsilon^{-1}}, |\psi\rangle, z)\}_{\lambda \in \mathbb{N}} \equiv \{\mathsf{Exp}_{\bot}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)\}_{\lambda \in \mathbb{N}}.$$

Proof of Lem. 24. Since $\mathsf{Exp}_{\bot}$ is efficient (because it never outputs s^*), $\mathcal{SE}_{\bot}$ just needs to run $\mathsf{Exp}_{\bot}$. □

Lemma 25 (Extract-and-Simulate for the Case of $b = \top$). *There is a QPT algorithm $\mathcal{SE}_{\top}$ such that for any polynomial-size quantum state $|\psi\rangle$ and noticeable ε,*

$$\{\mathcal{SE}_{\top}(1^{\lambda}, 1^{\varepsilon^{-1}}, |\psi\rangle, z)\}_{\lambda \in \mathbb{N}} \overset{s}{\approx}_{\varepsilon} \{\mathsf{Exp}_{\top}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)\}_{\lambda \in \mathbb{N}},$$

where $\gamma := \left(\frac{\varepsilon}{4}\right)^4$.

Proof of Lem. 25. Let δ be a noticeable function that satisfies Item 2 of Lem. 20 for $\gamma = \left(\frac{\varepsilon}{4}\right)^4$. We apply Lem. 22 where we define $\mathcal{A}_{\mathsf{CCY}}(\cdot)$ to be $\mathcal{K}(1^{\lambda}, 1^{\gamma^{-1}}, \cdot, z)$ (where z is considered as a classical advice for $\mathcal{A}_{\mathsf{CCY}}(\cdot)$). Note that the assumption of Lem. 22 is satisfied by Item 1 of Lem. 20. Let $\mathsf{Ext}_{\mathsf{CCY}}$ be the corresponding extractor and $S_{<\delta}$ and $S_{\geq\delta}$ be the decomposition of the Hilbert space of $|\psi\rangle$ as in Lem. 22. Then, we construct the extractor $\mathcal{SE}_{\bot}$ for Lem. 25 as follows:

$\mathcal{SE}_{\top}(1^{\lambda}, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$:

1. Run $(b_{\mathsf{CCY}}, s_{\mathsf{Ext}}, |\psi_{\mathsf{CCY}}\rangle) \leftarrow \mathsf{Ext}_{\mathsf{CCY}}(1^{\delta^{-1}} \mathcal{A}_{\mathsf{CCY}}, |\psi\rangle)$.
2. If $b_{\mathsf{CCY}} = \bot$, output Fail and immediately halt.
3. Run $(b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^{\lambda}, 1^{\gamma^{-1}}, |\psi_{\mathsf{CCY}}\rangle, z)$.
   – If $b = \top$, output $(\rho_{\mathsf{out}}, s_{\mathsf{Ext}})$.
   – If $b = \bot$, output Fail.

We decompose $|\psi\rangle$ as $|\psi\rangle = |\psi_{<\delta}\rangle + |\psi_{\geq\delta}\rangle$ where $|\psi_{<\delta}\rangle \in S_{<\delta}$ and $|\psi_{\geq\delta}\rangle \in S_{\geq\delta}$. By Item 2a of Lem. 22, for any $|\psi_{\geq\delta}\rangle \in S_{\geq\delta}$, it holds that

$$\Pr\left[\begin{array}{l} b_{\mathsf{CCY}} = \top \ \wedge \ s_{\mathsf{Ext}} = s^* \ \wedge \\ \| |\psi_{\geq\delta}\rangle\langle\psi_{\geq\delta}| - |\psi_{\mathsf{CCY},\geq\delta}\rangle\langle\psi_{\mathsf{CCY},\geq\delta}| \|_{tr} \leq \mathsf{negl}(\lambda) \end{array} : (b_{\mathsf{CCY}}, s_{\mathsf{Ext}}, |\psi_{\mathsf{CCY},\geq\delta}\rangle) \leftarrow \mathsf{Ext}_{\mathsf{CCY}}(1^{\delta^{-1}}, \mathcal{A}_{\mathsf{CCY}}, |\psi_{\geq\delta}\rangle)\right]$$
$$= 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over the randomness of internal measurements by $\mathsf{Ext}_{\mathsf{CCY}}$. On the other hand, by Item 2b of Lem. 22, for any $|\psi_{<\delta}\rangle \in S_{<\delta}$, it holds that

$$\Pr\left[b_{\mathsf{CCY}} = \bot \ \vee \ |\psi_{\mathsf{CCY},<\delta}\rangle \in S_{<\delta} : (b_{\mathsf{CCY}}, s_{\mathsf{Ext}}, |\psi_{\mathsf{CCY},<\delta}\rangle) \leftarrow \mathsf{Ext}_{\mathsf{CCY}}(1^{\delta^{-1}}, \mathcal{A}_{\mathsf{CCY}}, |\psi_{<\delta}\rangle)\right] = 1.$$

Combining them, if we run $\mathsf{Ext}_{\mathsf{CCY}}(1^{\delta^{-1}}, \mathcal{A}_{\mathsf{CCY}}, |\psi\rangle)$ except for the measurements of b_{CCY} and s_{Ext} registers, with overwhelming probability over internal measurement results by $\mathsf{Ext}_{\mathsf{CCY}}$, the output has a negligible trace distance from a state of the form

$$\sum_{b'_{\mathsf{CCY}},s'} \left|b'_{\mathsf{CCY}}\right\rangle \left|s'\right\rangle \left|\psi_{\mathsf{CCY},<\delta,b'_{\mathsf{CCY}},s'}\right\rangle + |\top\rangle\,|s^*\rangle\,|\psi_{\geq\delta}\rangle$$

where $\left|\psi_{\mathsf{CCY},<\delta,\top,s'}\right\rangle \in S_{<\delta}$ for all s'. We apply Lem. 23 where we set $|\phi_{0,0}\rangle := |\top\rangle\,|s^*\rangle\,|\psi_{<\delta}\rangle$, $|\phi_{0,1}\rangle := |\top\rangle\,|s^*\rangle\,\left|\psi_{\geq\delta}\right\rangle$, $|\phi_{1,0}\rangle := \sum_{b'_{\mathsf{CCY}},s'} \left|b'_{\mathsf{CCY}}\right\rangle \left|s'\right\rangle \left|\psi_{\mathsf{CCY},<\delta,b'_{\mathsf{CCY}},s'}\right\rangle$, and $|\phi_{1,1}\rangle := |\phi_{0,1}\rangle = |\top\rangle\,|s^*\rangle\,\left|\psi_{\geq\delta}\right\rangle$, and F is a quantum algorithm that works as follows:

$F\left(\sum_{b_{\mathsf{CCY}},s} |b_{\mathsf{CCY}}\rangle\,|s\rangle\,\left|\psi_{b_{\mathsf{CCY}},s}\right\rangle\right)$: It measures the first and second registers to get b_{CCY} and s after which the third register collapses to $\left|\psi_{b_{\mathsf{CCY}},s}\right\rangle$. It outputs Fail if $b_{\mathsf{CCY}} = \bot$ and otherwise runs $(b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}(1^\lambda, 1^{\gamma^{-1}}, \left|\psi_{b_{\mathsf{CCY}},s}\right\rangle, z)$. If $b = \top$, it outputs s and ρ_{out}. Otherwise, it outputs Fail.

Then, for $|\phi_0\rangle := |\phi_{0,0}\rangle + |\phi_{0,1}\rangle$ and $|\phi_1\rangle := |\phi_{1,0}\rangle + |\phi_{1,1}\rangle$, for any distinguisher D, Lem. 23 gives the following:

$$|\Pr[D(F(|\phi_0\rangle\langle\phi_0|)) = 1] - \Pr[D(F(|\phi_1\rangle\langle\phi_1|)) = 1]| \leq 4\gamma'^{1/4}$$

where $\gamma' := \max\left\{\Pr\left[F\left(\frac{|\phi_{0,0}\rangle}{\||\phi_{0,0}\rangle\|}\right) \neq \mathsf{Fail}\right], \Pr\left[F\left(\frac{|\phi_{1,0}\rangle}{\||\phi_{1,0}\rangle\|}\right) \neq \mathsf{Fail}\right]\right\}$ if $\gamma' < 1/36$. By the definition of F, it is easy to see that $F(|\phi_0\rangle\langle\phi_0|)$ is distributed according to $\mathsf{Exp}_\top(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)$ and the distribution of $F(|\phi_1\rangle\langle\phi_1|)$ is negligibly close to $\mathcal{SE}_\top(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$ conditioned on the fixed internal measurement outcomes by $\mathsf{Ext}_{\mathsf{CCY}}$. Since this holds for overwhelming fraction of internal measurement outcomes by $\mathsf{Ext}_{\mathsf{CCY}}$, we have

$$\{\mathcal{SE}_\top(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)\}_{\lambda\in\mathbb{N}} \overset{\mathrm{s}}{\approx}_{4\gamma'^{1/4}} \{\mathsf{Exp}_\top(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)\}_{\lambda\in\mathbb{N}}. \tag{82}$$

By the definition of F, for any state of the form $|\bot\rangle\,|s\rangle\,|\psi\rangle$, we clearly have

$$\Pr[F(|\bot\rangle\,|s\rangle\,|\psi\rangle) \neq \mathsf{Fail}] = 0. \tag{83}$$

Since $|\psi_{<\delta}\rangle \in S_{<\delta}$ and $\left|\psi_{\mathsf{CCY},<\delta,\top,s'}\right\rangle \in S_{<\delta}$ for all s', by Item 1 of Lem. 22, we have

$$\Pr\left[\mathcal{A}_{\mathsf{CCY}}\left(\frac{|\psi_{<\delta}\rangle}{\||\psi_{<\delta}\rangle\|}\right) = s^*\right] < \delta$$

and

$$\Pr\left[\mathcal{A}_{\mathsf{CCY}}\left(\frac{\left|\psi_{\mathsf{CCY},<\delta,\top,s'}\right\rangle}{\|\left|\psi_{\mathsf{CCY},<\delta,\top,s'}\right\rangle\|}\right) = s^*\right] < \delta$$

for any s'.

Recalling that $\mathcal{A}_{\mathsf{CCY}}(\cdot) = \mathcal{K}(1^\lambda, 1^{\gamma^{-1}}, \cdot)$, by the contraposition of Item 2 of Lem. 20, the above inequalities imply

$$\Pr\left[b = \top : (b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}\left(1^\lambda, 1^{\gamma^{-1}}, \frac{|\psi_{<\delta}\rangle}{\||\psi_{<\delta}\rangle\|}, z\right)\right] < \gamma \tag{84}$$

and

$$\Pr\left[b = \top : (b, \rho_{\mathsf{out}}) \leftarrow \mathcal{G}\left(1^\lambda, 1^{\gamma^{-1}}, \frac{\left|\psi_{\mathsf{CCY},<\delta,\top,s'}\right\rangle}{\|\left|\psi_{\mathsf{CCY},<\delta,\top,s'}\right\rangle\|}, z\right)\right] < \gamma \tag{85}$$

for any s'.

By Eq. (83) to (85) and the definitions of F, $|\phi_{0,0}\rangle$, $|\phi_{1,0}\rangle$, and γ', we can see that $\gamma' < \gamma = \left(\frac{\varepsilon}{4}\right)^4$ and thus $\gamma < 1/36$ holds for a sufficiently large security parameter λ. By plugging $4\gamma'^{1/4} < \varepsilon$ into Eq. (82), we complete the proof of Lem. 25.

□

Given Lem. 24 and Lem. 25, the rest of the proof of Lem. 20 is very similar to the corresponding part of the ε-zero-knowledge property of the protocols in [CCY21]. We give the full proof for completeness.

Let $\mathcal{SE}_{\mathsf{comb}}$ be an algorithm that works as follows:

$\mathcal{SE}_{\mathsf{comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$:

1. Set $\varepsilon' := \frac{\varepsilon^2}{4\log^4(\lambda)}$.
2. Choose $\mathsf{mode} \leftarrow \{\top, \bot\}$.
3. Run and output $\mathcal{SE}_{\mathsf{mode}}(1^\lambda, 1^{\varepsilon'^{-1}}, |\psi\rangle)$.

Lemma 26 **($\mathcal{SE}_{\mathsf{comb}}$ Simulates Exp with Probability almost $1/2$).** *Let $p^{\mathsf{suc}}_{\mathsf{comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$ be the probability that $\mathcal{SE}_{\mathsf{comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$ does not return* Fail, *and let*

$$D_{\mathsf{ext,comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$$

be a conditional distribution of $\mathcal{SE}_{\mathsf{comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$, conditioned on that it does not return Fail. *Then we have*

$$\left|p^{\mathsf{suc}}_{\mathsf{comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z) - 1/2\right| \leq \varepsilon'/2 + \mathsf{negl}(\lambda). \tag{86}$$

Moreover, we have

$$\{D_{\mathsf{ext,comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)\}_{\lambda\in\mathbb{N}} \overset{s}{\approx}_{4\varepsilon'} \{\mathsf{Exp}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)\}_{\lambda\in\mathbb{N}}, \tag{87}$$

where $\gamma := \left(\frac{\varepsilon'}{4}\right)^4$.

Proof. (sketch.) The intuition behind this proof is as follows. By Lem. 24 and 25, $\mathcal{SE}_\bot$ and $\mathcal{SE}_\top$ almost simulate Exp conditioned on that $b = \bot$ and $b = \top$, respectively. Therefore, if we randomly guess b and runs either of $\mathcal{SE}_\bot$ or $\mathcal{SE}_\top$ that successfully works for the guessed case, the output distribution is close to the real output distribution of Exp conditioned on that the guess is correct, which happens with probability almost $1/2$.

A formal proof can be obtained based on the above intuition and is exactly the same as the proof of [CCY21, Lemma 5.5] except for notational adaptations.

□

Then, we convert $\mathcal{SE}_{\mathsf{comb}}$ into a full-fledged simulator that does not return Fail by using Watrous' rewinding lemma (Lem. 21). Namely, we let Q be a quantum algorithm that takes $|\psi\rangle$ as input and outputs $\mathcal{SE}_{\mathsf{comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$ where $b := 0$ if and only if it does not return Fail, $p_0 := \frac{1}{4}$, $q := \frac{1}{2}$, $\alpha := \varepsilon'$, and $T := 2\log(1/\varepsilon')$. Then it is easy to check that the conditions for Lem. 21 is satisfied by Inequality (86) in Lem. 26 (for sufficiently large λ). Then, by using Lem. 21, we can see that $\mathsf{R}(1^T, \mathsf{Q}, |\psi\rangle)$ runs in time $T = \mathsf{poly}(\lambda)$ and its output (seen as a mixed state) has a trace distance bounded by $4\sqrt{\alpha}\frac{\log(1/\alpha)}{p_0(1-p_0)}$ from $D_{\mathsf{ext,comb}}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)$. Since we have $\alpha = \varepsilon' =$

$\frac{\varepsilon^2}{4\log^4(\lambda)} = 1/\mathsf{poly}(\lambda)$, we have $4\sqrt{\alpha}\frac{\log(1/\alpha)}{p_0(1-p_0)} < \sqrt{\alpha}\log^2(\lambda) = \frac{\varepsilon}{2}$ for sufficiently large λ where we used $\log(1/\alpha) = \log(\mathsf{poly}(\lambda)) = o(\log^2(\lambda))$ and $\frac{4}{p_0(1-p_0)} = O(1)$. Thus, by combining the above and Eq. (87) in Lem. 26, if we define $\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z) := \mathsf{R}(1^T, \mathsf{Q}, |\psi\rangle)$, then we have

$$\{\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, |\psi\rangle, z)\}_{\lambda\in\mathbb{N}} \overset{\mathrm{s}}{\approx}_{\frac{\varepsilon}{2}+4\varepsilon'} \{\mathsf{Exp}(\lambda, 1^{\gamma^{-1}}, |\psi\rangle, z)\}_{\lambda\in\mathbb{N}}$$

where $\gamma = \left(\frac{\varepsilon'}{4}\right)^4 = \mathsf{poly}(\varepsilon)$. We can conclude the proof of Lem. 20 by noting that we have $\frac{\varepsilon}{2}+4\varepsilon' < \varepsilon$ since we have $\varepsilon' = \frac{\varepsilon^2}{4\log^4(\lambda)} < \frac{\varepsilon}{8}$ for sufficiently large λ.

B Small-Tag, One-sided, Synchronous, Post-Quantum Setting

B.1 Construction

In this section, we prove that Prot. 1 is post-quantumly secure if we rely on post-quantum building-blocks. Similarly to the classical case in Sec. 4, we present the construction assuming the existence of post-quantum *injective* OWFs. The assumption can be relaxed to *any* OWFs by an appropriate modification to the protocol similarly to the classical case. Since this is exactly the same as that in the classical setting (i.e., Sec. 4.8), we do not repeat it in this section.

Our construction is based on the following building blocks:

– A post-quantum *injective* OWF f;

– Naor's commitment Com that is implemented with a post-quantum OWF.[37];

– A post-quantum witness-indistinguishable argument of knowledge with ε-close emulation WIAoK (as per Def. 14).

Protocol 5: Small-Tag One-Sided Synchronous Post-Quantum NMCom $\langle C, R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$

The tag space is defined to be $[n]$ where n is a polynomial on λ. Let $t \in [n]$ be the tag for the following interaction. Let m be the message to be committed to.

Commit Stage:

1. Receiver R samples and sends the first message β for Naor's commitment;
2. Committer C commits to m using the second message of Naor's commitment. Formally, C samples a random tape r and sends $\mathsf{com} = \mathsf{Com}_\beta(m; r)$;
3. R computes $\{y_i = f(x_i)\}_{i\in[t]}$ with $x_i \xleftarrow{\$} \{0,1\}^\lambda$ for each $i \in [t]$. R sends $Y = (y_1, \ldots, y_t)$ to C;
4. **(WIAoK-1.)** R and C execute an instance of WIAoK where R proves to C that he "knows" a pre-image of some y_i contained in Y (defined in Step 3). Formally, R proves that $Y \in \mathcal{L}^t_f$, where
$$\mathcal{L}^t_f := \{(y_1, \ldots, y_t) \mid \exists (i, x_i) \ s.t.\ i \in [t] \wedge y_i = f(x_i)\}. \tag{88}$$
Note that R uses $(1, x_1)$ as the witness when executing this WIAoK.
5. **(WIAoK-2.)** C and R execute an instance of WIAoK where C proves to R that he "knows" *either* the message committed in com (defined in Step 2), *or* a pre-image of some y_i contained in Y (defined in Step 3). Formally, C proves that $(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$, where $\mathcal{L}_\beta \vee \mathcal{L}^t_f$ denotes

[37] In this way, the Naor's commitment is statistically binding and *post-quantumly* computationally hiding.

the OR-composed language (as per Def. 1), $\mathcal{L}_f^t$ was defined in Language (88) and

$$\mathcal{L}_\beta := \{\mathsf{com} \mid \exists (m,r) \ s.t. \ \mathsf{com} = \mathsf{Com}_\beta(m;r)\}. \tag{89}$$

Note that C uses the (m,r) defined in Step 2 as the witness when executing this WIAoK.

Decommit Stage: C sends (m,r). R accepts if $\mathsf{com} = \mathsf{Com}_\beta(m;r)$, and rejects otherwise.

Remark that the above protocol is exactly the same as Prot. 1 except that we use post-quantum WIAoK with ε-close emulation and assume post-quantum security for all other building-blocks.

Security. Completeness is straightforward from the description of Prot. 5. The statistical binding property follows from that of Naor's commitment. Computational hiding of any non-malleable commitment follows from its non-malleability. So, we only need to show that Prot. 5 is non-malleable. This is established by the following Thm. 29, which we prove in subsequent subsections.

Theorem 29. *The commitment scheme $\langle C,R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$ in Prot. 5 is non-malleable against one-sided synchronous QPT adversaries with tag space $[n]$, with n being any polynomial on λ.*

B.2 Proving Non-Malleability

We prove Thm. 29 in Appx. B.2 to B.7. The proof follows the same template used in Sec. 4.2. The only exception is the proof of Lem. 28 (which is the post-quantum counterpart of Lem. 7). Recall that in Lem. 7, we amplify the extractor $\mathcal{K}$ to the simulation-extractor $\mathcal{SE}$ by rewinding. In the current post-quantum setting, we cannot rewind quantum algorithms in general. Therefore, we rely on an alternative argument based on our new extract-and-simulate lemma (Lem. 20). The rest of the proof is almost the same as its classical counterpart; Thus, many parts of the proof are taken verbatim from there. Essentially, the only difference is that we have to deal with noticeable errors that come from witness-extended ε-close emulator (as per Property 2 of Def. 14). We highlight differences from the classical counterpart in purple color throughout this section.

We use the same notation as in the proof of Thm. 2, with the only difference that now the adversaries are non-uniform QPT (instead of PPT) machines. It is worth noting that the honest committer and receiver are still classical (i.e., non-uniform PPT) machines.

In the sequel, we write $\langle C,R\rangle$ to mean $\langle C,R\rangle^{\mathsf{OneSided}}_{\mathsf{tg},\mathsf{PQ}}$ for notational convenience.

Game $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$: This game is identical to its classical counterpart defined on Page 26, except that $\mathcal{M} = \{\mathcal{M}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$ now is a (non-uniform) QPT machine. That is, this is the man-in-the-middle execution of the commit stage of the $\langle C,R\rangle$ defined in Prot. 5, where the left committer commits to m and $\mathcal{M}$'s non-uniform advice is ρ_λ. The output of $H^{\mathcal{M}_\lambda}(m,\rho_\lambda)$ is again defined to be $(\mathsf{OUT}_\mathcal{M}, \widetilde{\tau}, b)$, where $\mathsf{OUT}_\mathcal{M}$ is the (quantum) output of $\mathcal{M}_\lambda(\rho_\lambda)$ at the end of this game, $\widetilde{\tau}$ consists of the Steps 1 and 2 messages exchanged in the right session, and $b \in \{\top, \bot\}$ is the honest receiver's final decision.

Notation. Recall from Def. 7 that the man-in-the middle game for a QPT adversary $\mathcal{M}$ is denoted by $\overline{\mathsf{mim}}^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m,\rho_\lambda)$. Also, note that "$\overset{\mathrm{c}}{\approx}$" refers to *quantumly* computational indistinguishability throughout this section.

For any $(\mathsf{OUT}_{\mathcal{M}}, \widetilde{\tau}, b)$ in the support of $H^{\mathcal{M}_\lambda}(\lambda, m, z)$, we define $\mathsf{val}_b(\widetilde{\tau})$ similarly to the classical case, i.e.,

$$\mathsf{val}_b(\widetilde{\tau}) \coloneqq \begin{cases} \mathsf{val}(\widetilde{\tau}) & b = \top \\ \bot & b = \bot \end{cases},$$

where $\mathsf{val}(\widetilde{\tau})$ denote the value statistically-bound in Steps 1 and 2 of the right session.

Similarly to Eq. (13), we have

$$\left\{\overline{\mathsf{mim}}^{\mathcal{M}_\lambda}_{\langle C,R\rangle}(m, \rho_\lambda)\right\} \overset{\text{i.d.}}{=\!=} \left\{\big(\mathsf{OUT}, \mathsf{val}_b(\widetilde{\tau})\big) : (\mathsf{OUT}, \widetilde{\tau}, b) \leftarrow H^{\mathcal{M}_\lambda}(m, \rho_\lambda)\right\}, \tag{90}$$

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

Then, similarly to Eq. (14), the post-quantum non-malleability can be reduced to establishing the following equation:

$$\begin{aligned} &\left\{\big(\mathsf{OUT}^0, \mathsf{val}_{b^0}(\widetilde{\tau}^0)\big) : (\mathsf{OUT}^0, \widetilde{\tau}^0, b^0) \leftarrow H^{\mathcal{M}_\lambda}(m_0, \rho_\lambda)\right\} \\ \overset{\text{c}}{\approx}\ &\left\{\big(\mathsf{OUT}^1, \mathsf{val}_{b^1}(\widetilde{\tau}^1)\big) : (\mathsf{OUT}^1, \widetilde{\tau}^1, b^1) \leftarrow H^{\mathcal{M}_\lambda}(m_1, \rho_\lambda)\right\}, \end{aligned} \tag{91}$$

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 by Contradiction. Similarly to Inequality (15), we assume for contradiction that there are a (possibly non-uniform) QPT distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda, \sigma_\lambda\}_{\lambda\in\mathbb{N}}$ and a function $\delta(\lambda) = 1/\mathsf{poly}(\lambda)$ such that for infinitely many $\lambda \in \mathbb{N}$, it holds that

$$\left|\Pr\left[\mathcal{D}_\lambda\big(\mathsf{OUT}^0, \mathsf{val}_{b^0}(\widetilde{\tau}^0); \sigma_\lambda\big) = 1\right] - \Pr\left[\mathcal{D}_\lambda\big(\mathsf{OUT}^1, \mathsf{val}_{b^1}(\widetilde{\tau}^1); \sigma_\lambda\big) = 1\right]\right| \geq 3 \cdot \delta(\lambda), \tag{92}$$

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

Then, we show the following Lem. 27, which should be understood as the post-quantum counterpart of Lem. 6.

Lemma 27. *For the above $\delta(\lambda)$, there exits a hybrid G such that for any QPT $\mathcal{M} = \{\mathcal{M}_\lambda, \rho_\lambda\}_{\lambda\in\mathbb{N}}$, the following holds*

1. $\left\{(\mathsf{OUT}^0, \mathsf{Val}^0) : (\mathsf{OUT}^0, \mathsf{Val}^0) \leftarrow G^{\mathcal{M}_\lambda}(\lambda, m_0, \rho_\lambda)\right\} \overset{c}{\approx} \left\{(\mathsf{OUT}^1, \mathsf{Val}^1) : (\mathsf{OUT}^1, \mathsf{Val}^1) \leftarrow G^{\mathcal{M}_\lambda}(\lambda, m_1, \rho_\lambda)\right\}$, *where both ensembles are indexed by $\lambda \in \mathbb{N}$ and $(m_0, m_1) \in \{0,1\}^{\ell(\lambda)} \times \{0,1\}^{\ell(\lambda)}$.*
2. $\left\{(\mathsf{OUT}^G, \mathsf{Val}^G) : (\mathsf{OUT}^G, \mathsf{Val}^G) \leftarrow G^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)\right\} \overset{c}{\approx}_{\delta(\lambda)} \left\{(\mathsf{OUT}^H, \mathsf{val}_{b^H}(\widetilde{\tau}^H)) : (\mathsf{OUT}^H, \widetilde{\tau}^H, b^H) \leftarrow H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)\right\}$, *where both ensembles are indexed by $\lambda \in \mathbb{N}$ and $m \in \{0,1\}^{\ell(\lambda)}$.*

It is easy to see that if Lem. 27 is true, it contradicts our assumption in Inequality (92). Therefore, it will finish the proof of non-malleability. Indeed, this lemma is the most technically involved part. We prove it in Appx. B.3.

B.3 Proof of Lem. 27

Similarly to Algo. 4.1, we provide a new but equivalent interpretation of the game $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ in Algo. B.1. We also provide a picture in Fig. 11a to illustrate it.

Algorithm B.1: Re-interpretation of Game $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$

Game $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ can be split into the following stages:

1. **Prefix Generation:** First, execute Steps 1 and 2 of the man-in-the-middle game of Prot. 5. That is, it plays as the left honest committer committing to m and the right honest receiver, with $\mathcal{M}_\lambda(\rho_\lambda)$ being the man-in-the-middle adversary.

   Notation: Let $\mathsf{st}_\mathcal{M}$ denote the state of $\mathcal{M}$ at the end of Step 2; Let st_C (resp. st_R) denote the state of the honest committer (resp. receiver) at the end of Step 2; Let $\widetilde{\tau}$ denote the tuple $(\widetilde{\beta}, \widetilde{\mathsf{com}})$[a]. In terms of notation, we denote the execution of this stage by

   $$(\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau}) \leftarrow H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda). \tag{93}$$

   We will call the tuple $(\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ the *prefix* and denote it by pref. It is worth noting that this pref contains all the information such that a QPT machine can "complete" the remaining execution of $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ starting from pref.

2. **The Remainder:** Next, it simply resumes from where the **Prefix Generation** stage stops, to finish the remaining steps of the man-in-the-middle execution $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$.

   Notation: We introduce the following notations to describe this stage. Define a QPT machine $\mathcal{A}$ that takes as input $(\mathsf{st}_\mathcal{M}, \widetilde{\tau})$; Machine $\mathcal{A}_\lambda$ is supposed to run the residual strategy of $\mathcal{M}_\lambda$ starting from $\mathsf{st}_\mathcal{M}$. Also, define a QPT machine $\mathcal{B}$ that takes as input $(\mathsf{st}_C, \mathsf{st}_R, \widetilde{\tau})$; Machine $\mathcal{B}$ is supposed to run the residual strategies of the honest committer C and receiver R, starting from st_C and st_R respectively[b]. With the above notations, we can denote the execution of the remaining steps of $H^{\mathcal{M}}(\lambda, m, z)$ by

   $$(\mathsf{OUT}_\mathcal{A}, b) \leftarrow \langle \mathcal{A}_\lambda(\mathsf{st}_\mathcal{M}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau}), \tag{94}$$

   where $\mathsf{OUT}_\mathcal{A}$ is the output of $\mathcal{A}_\lambda$, and $b \in \{\bot, \top\}$ is the output of the honest receiver R (in the right), indicating if the man-in-the-middle's commitment (i.e., the right session) is accepted ($b = \top$) or not ($b = \bot$). (We remark that $\mathsf{OUT}_\mathcal{A}$ is nothing but the man-in-the-middle $\mathcal{M}$'s final output.)

3. **Output:** It outputs the tuple $(\mathsf{OUT}_\mathcal{A}, \widetilde{\tau}, b)$.

[a] Recall that $\widetilde{\beta}$ and $\widetilde{\mathsf{com}}$ are the Steps 1 and 2 messages in the right session; they constitutes an execution of Naor's commitment.

[b] Note that it is not necessary to give $\widetilde{\tau}$ as common input to these parties; indeed, it can be included in their respective internal states. We choose to make $\widetilde{\tau}$ explicit only to match the syntax of Lem. 20.

We prove the following lemma.

Lemma 28. *Let $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$, $\mathcal{A}_\lambda$, and $\mathcal{B}$ be as defined in Algo. B.1. There exists a QPT machine $\mathcal{SE}$ (the simulation-extractor) such that for any $(\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ in the support of $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$, any noticeable $\varepsilon(\lambda)$, it holds that*

$$\begin{aligned} &\big\{(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}) : (\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}) \leftarrow \mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \mathcal{A}_\lambda, \mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}} \\ \overset{c}{\approx}_{\varepsilon(\lambda)} \ &\big\{\big(\mathsf{OUT}_\mathcal{A}, \mathsf{val}_b(\widetilde{\tau})\big) : (\mathsf{OUT}_\mathcal{A}, b) \leftarrow \langle \mathcal{A}_\lambda(\mathsf{st}_\mathcal{M}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}} \end{aligned}$$

Remark 18. Compared to the classical counterpart (Lem. 7), $\mathcal{SE}$ takes $\mathcal{A}_\lambda$ and $\mathsf{st}_\mathcal{M}$ as part of its input instead of accessing $\mathcal{A}_\lambda(\mathsf{st}_\mathcal{M})$. Though we can see that $\mathcal{SE}$ actually makes only black-box use of $\mathcal{A}_\lambda(\mathsf{st}_\mathcal{M})$

in a certain sense, we do not try to formally state it because black-box simulation is not our focus. Another minor difference is that we removed $b_{\mathcal{SE}}$ from the output of $b_{\mathcal{SE}}$. This is because $b_{\mathcal{SE}}$ was only used in the proof of the classical counterpart of Lem. 28 (i.e., Lem. 7), which will be replaced with a different proof based on Lem. 20.

Lem. 28 is the main technical lemma for the current proof of Lem. 27. We present its proof in Appx. B.4. In the following, we finish the proof of Lem. 27 assuming that Lem. 28 is true.

With Lem. 28, we are now ready to present the description of G.

Algorithm B.2: Hybrid $G^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$

This hybrid proceeds as follows:

1. **Prefix Generation:** This stage is identical to Stage 1 of $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$. Formally, it executes
$$(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau}) \leftarrow H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda),$$
where $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$ is defined in Stage 1 of Algo. B.1.
2. **The Remainder:** Define $\mathcal{A}_\lambda$ in the same way as in Stage 2 of $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$. With this $\mathcal{A}_\lambda$ and the $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ from the previous stage, $G^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ invokes the $\mathcal{SE}$ prescribed in Lem. 28. Formally, it executes the following procedure:
$$(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}}) \leftarrow \mathcal{SE}(1^\lambda, 1^{\delta^{-1}}, \mathcal{A}_\lambda, \mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau}),$$
where the δ is the statistical distance that we want to show for Property 2 of Lem. 27.

   Remark 19. We emphasize that in this stage, $G^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ does *not* make use of st_C.
3. **Output:** It outputs $(\mathsf{OUT}_{\mathcal{SE}}, \mathsf{Val}_{\mathcal{SE}})$.

Proving Property 1 of Lem. 27. Observe that hybrid $G^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ is an efficient machine, since both $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$ and $\mathcal{SE}(1^\lambda, 1^{\delta^{-1}}, \mathcal{A}_\lambda, \mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ are efficient. Moreover, it does not rewind Steps 1 and 2 of the man-in-the-middle execution, and $\mathcal{SE}(1^\lambda, 1^{\delta^{-1}}, \mathcal{A}_\lambda, \mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ does *not* need to know st_C. Therefore, Property 1 of Lem. 27 follows immediately from the computational-hiding property of the left Naor's commitment (i.e., Steps 1 and 2 in the left session).

Proving Property 2 of Lem. 27. First, observe that the distribution of the prefix is identical in $G^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ and $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$. For each fixed prefix, Lem. 28 implies that $G^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ and $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ are $\delta(\lambda)$-computationally indistinguishable (notice that $G^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ runs $\mathcal{SE}$ with the second input $1^{\delta^{-1}}$). This immediately implies Property 2 of Lem. 27.

B.4 Proof of Lem. 28

First, we describe a machine $\mathcal{G}_1$ that simulates the real execution without using st_C. Unlike its classical counterpart, $\mathcal{G}_1$ is parameterized by a noticeable function $\varepsilon_{\mathsf{wiaok}}(\lambda)$, which is used as the error parameter for witness-extended emulators for WIAoK (as per Property 2 of Def. 14).

Machine $\mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}]$: (Illustrated in Fig. 11b) For any prefix pref, $\mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to Stage 2 of $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ shown in Algo. B.1 (and depicted in Fig. 11a), except for the following difference: Instead of executing the left **WIAoK-1** honestly, it uses the witness-extended

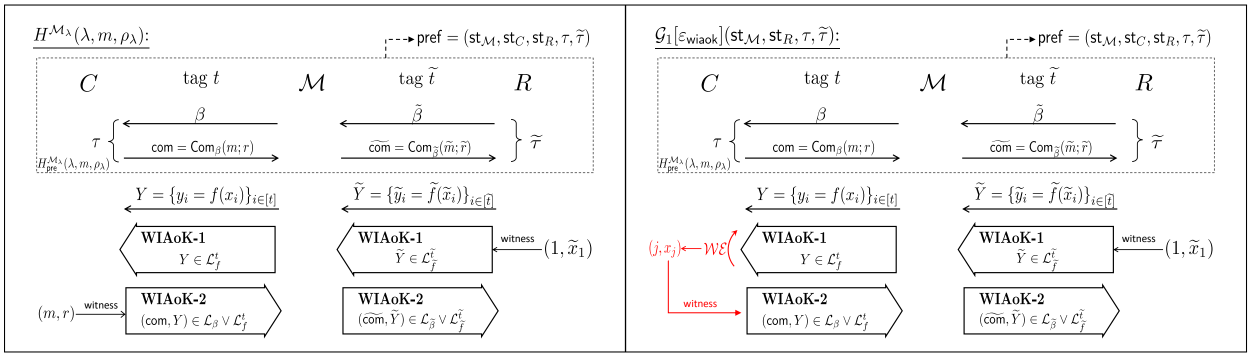

$H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$:
pref = $(\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\widetilde{\beta}$
τ
$\widetilde{\tau}$
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
$H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$
$Y = \{y_i = f(x_i)\}_{i\in[t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i\in[\widetilde{t}]}$
WIAoK-1
$Y \in \mathcal{L}^t_f$
$\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
witness
$(1, \widetilde{x}_1)$
WIAoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
(m, r)
$\mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
$(j, x_j) \leftarrow \mathcal{WE}$
(a)
(b)

Fig. 11: Machines $H^{\mathcal{M}}$ and $\mathcal{G}_1$ (Difference is highlighted in red color)

emulator (as per Property 2 of Def. 14) of the left **WIAoK-1** with the error parameter $\varepsilon_{\mathsf{wiaok}}(\lambda)$ to extract a witness, and

– If the left committer accepts the left **WIAoK-1** and the extracted witness is valid (i.e., it is a (j, x_j) pair such that $y_j = f(x_j)$ for some $j \in [t]$), $\mathcal{G}_1$ uses (j, x_j) to finish the left **WIAoK-2**. Similarly to $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$, $\mathcal{G}_1$ eventually outputs $\mathcal{M}$'s final state and the right receiver's decision bit b;

– If the left committer accepts the left **WIAoK-1** but the extracted witness is invalid, it aborts immediately and outputs $(\bot, \bot)$.

– If the left committer rejects the left **WIAoK-1**, it runs the rest of execution of $H^{\mathcal{M}_\lambda}(\lambda, m, \rho_\lambda)$ to output $\mathcal{M}$'s final state and the right receiver's decision bit b. Note that it does not need to run the left **WIAoK-2** in this case since the left committer aborts after the left **WIAoK-1**.

We denote the above procedure by $(\mathsf{OUT}, b) \leftarrow \mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$. It is worth noting that $\mathcal{G}_1$ does *not* need to know st_C.

Next, we prove a lemma that shows that $\mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}]$ simulates the real execution up to an error $\varepsilon_{\mathsf{wiaok}}$.

Lemma 29. *Let $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$, $\mathcal{A}_\lambda$, and $\mathcal{B}$ be as defined in Algo. B.1. For any $\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ in the support of $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$, it holds that*

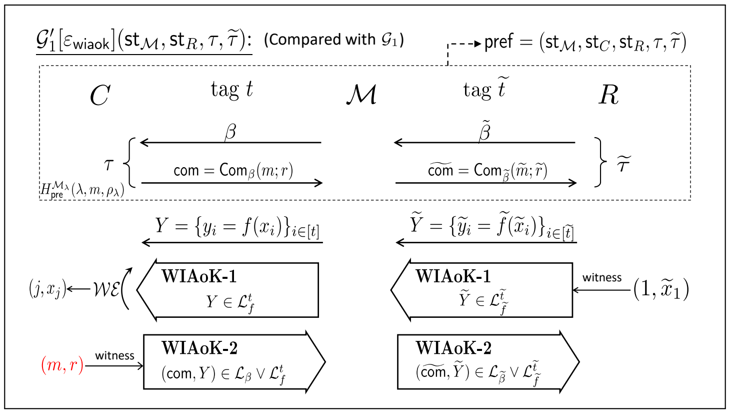
$$\big\{\big(\mathsf{OUT},\ b\big) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}}$$
$$\overset{c}{\approx}_{\varepsilon_{\mathsf{wiaok}}(\lambda)} \big\{\big(\mathsf{OUT}_\mathcal{A},\ b\big) : (\mathsf{OUT}_\mathcal{A}, b) \leftarrow \langle \mathcal{A}_\lambda(\mathsf{st}_\mathcal{M}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}}.$$

Proof. We first define an intermediate machine $\mathcal{G}'_1[\varepsilon_{\mathsf{wiaok}}]$ below.

Machine $\mathcal{G}'_1[\varepsilon_{\mathsf{wiaok}}]$: (Illustrated in Fig. 12) For any prefix pref, $\mathcal{G}'_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it uses (m, r) as the witness in the left **WIAoK-2** even if it succeeds in extracting a valid witness from the left **WIAoK-1**.

By the WI property of the left **WIAoK-2**, it holds that

$$\begin{aligned} &\big\{\big(\mathsf{OUT},\ b\big) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}} \\ \overset{c}{\approx}\ &\big\{\big(\mathsf{OUT},\ b\big) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}'_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda\in\mathbb{N}} \end{aligned}. \quad (95)$$



$\mathcal{G}'_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$: (Compared with $\mathcal{G}_1$)
pref = $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\widetilde{\beta}$
τ
com = $\mathsf{Com}_{\beta}(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
$\widetilde{\tau}$
$H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$
$Y = \{y_i = f(x_i)\}_{i \in [t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i \in [\widetilde{t}]}$
$(j, x_j) \leftarrow \mathcal{WE}$
WIAoK-1
$Y \in \mathcal{L}^t_f$
WIAoK-1
$\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
witness
$(1, \widetilde{x}_1)$
(m, r)
witness
WIAoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_{\beta} \vee \mathcal{L}^t_f$
WIAoK-2
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$

Fig. 12: Machine $\mathcal{G}'_1$ (Difference with $\mathcal{G}_1$ is highlighted in red color)

Note that the only difference between $\mathcal{G}'_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ and $\langle \mathcal{A}_\lambda(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})$ is that the former runs the witness-extended emulator of the left **WIAoK-1** with the error parameter $\varepsilon_{\mathsf{wiaok}}(\lambda)$ (but does not use the extracted witness at all). Thus, by the AoK property of the left **WIAoK-1**, it holds that

$$\begin{aligned} &\big\{(\mathsf{OUT},\ b) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}'_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big\}_{\lambda \in \mathbb{N}} \\ \overset{\text{c}}{\approx}_{\varepsilon_{\mathsf{wiaok}}(\lambda)} &\big\{(\mathsf{OUT}_{\mathcal{A}},\ b) : (\mathsf{OUT}_{\mathcal{A}}, b) \leftarrow \langle \mathcal{A}_\lambda(\mathsf{st}_{\mathcal{M}}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})\big\}_{\lambda \in \mathbb{N}} \end{aligned}. \tag{96}$$

By combining Eq. (95) and (96), we obtain Lem. 29.

□

Next, we define the probability of R being convinced in the execution of $\mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}]$. This value plays an important role later in our proof.

Definition 16. *For any* $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$ *in the support of* $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$ *and any noticeable* $\varepsilon_{\mathsf{wiaok}}$*, we define the following value* $p^{\mathsf{Sim}}_{\mathsf{pref}}[\varepsilon_{\mathsf{wiaok}}]$*:*

$$p^{\mathsf{Sim}}_{\mathsf{pref}}[\varepsilon_{\mathsf{wiaok}}] \coloneqq \Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})]$$

Next, we show a technical lemma that gives an extractor $\mathcal{K}$ without the simulation property.

Lemma 30. *Let* $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$*,* $\mathcal{A}_\lambda$*,* $\mathcal{B}$ *be as defined in Algo. B.1. There exists a QPT machine* $\mathcal{K}$ *such that for any noticeable* $\varepsilon(\lambda)$*, there is a noticeable* $\varepsilon_{\mathsf{wiaok}}(\lambda)$ *that is efficiently computable from* $\varepsilon(\lambda)$ *such that the following holds for any* $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$*:*

1. **(Syntax.)** $\mathcal{K}$ *takes as input* $(1^\lambda, 1^{\varepsilon^{-1}}, \mathcal{A}_\lambda, \mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$*. It outputs a value* $\mathsf{Val}_{\mathcal{K}} \in \{0,1\}^{\ell(\lambda)} \cup \{\bot\}$ *such that* $\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau})$ *whenever* $\mathsf{Val}_{\mathcal{K}} \neq \bot$*.*
2. *If* $p^{\mathsf{Sim}}_{\mathsf{pref}}[\varepsilon_{\mathsf{wiaok}}] \geq \varepsilon(\lambda)$*, then it holds that*

$$\Pr\Big[\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val}_{\mathcal{K}} \leftarrow \mathcal{K}(1^\lambda, 1^{\varepsilon^{-1}}, \mathcal{A}_\lambda, \mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\Big] \geq \frac{\varepsilon'(\lambda)}{\widetilde{t}},$$

where $\varepsilon'(\lambda) \coloneqq \frac{\varepsilon(\lambda)}{10t^2}$*.*

Remark 20. Compared to the classical counterpart (Lem. 9), $\mathcal{K}$ takes $1^{\varepsilon^{-1}}$ as an additional input. This is needed because we only assume the AoK property via witness-extended ε-*close* emulation (as per Property 2 of Def. 14). As a positive effect of the above difference, $\mathcal{K}$ in Lem. 30 runs in *strict* QPT whereas it runs in *expected* PPT in the classical counterpart (Lem. 9). Also, we give $\mathcal{A}_\lambda$ and $\mathsf{st}_\mathcal{M}$ as part of input to $\mathcal{K}$ instead of giving oracle access to $\mathcal{A}_\lambda(\mathsf{st}_\mathcal{M})$ for a similar reason in Rmk. 18.

We will prove Lem. 30 in Appx. B.5. In the rest of this subsection, we finish the proof of Lem. 28 assuming Lem. 30 is true. This is the only part of the proof of Thm. 29 that significantly differs from its classical counterpart.

Proof of Lem. 28. We apply Lem. 20 where $\mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}]$, $\mathcal{K}$, $(\mathcal{A}_\lambda, \mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau)$, $\widetilde{\tau}$, and $\mathsf{val}(\widetilde{\tau})$ play the roles of $\mathcal{G}$, $\mathcal{K}$, ρ_λ, z_λ, and $s^*_{z_\lambda}$ in Lem. 20, respectively. Then, Lem. 30 ensures that the assumptions of Lem. 20 are satisfied.[38] Thus, there exists a polynomial poly and a QPT machine $\mathcal{SE}'$ such that for any noticeable function $\overline{\varepsilon}(\lambda)$,

$$\begin{aligned} &\left\{(\mathsf{OUT}_{\mathcal{SE}'}, \mathsf{Val}_{\mathcal{SE}'}) : (\mathsf{OUT}_{\mathcal{SE}'}, \mathsf{Val}_{\mathcal{SE}'}) \leftarrow \mathcal{SE}'(1^\lambda, 1^{\overline{\varepsilon}^{-1}}, \mathcal{A}_\lambda, \mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}} \\ \overset{\mathrm{s}}{\approx}_{\overline{\varepsilon}(\lambda)} &\left\{\left(\mathsf{OUT}, \mathsf{val}_b(\widetilde{\tau})\right) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}} \end{aligned} \quad (97)$$

for $\varepsilon_{\mathsf{wiaok}}(\lambda) = \mathsf{poly}(\overline{\varepsilon}(\lambda))$.

Moreover, since $\mathsf{val}_b(\widetilde{\tau})$ is determined by b for each fixed $\widetilde{\tau}$, Lem. 29 directly implies

$$\begin{aligned} &\left\{\left(\mathsf{OUT},\ \mathsf{val}_b(\widetilde{\tau})\right) : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}](\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}} \\ \overset{\mathrm{c}}{\approx}_{\varepsilon_{\mathsf{wiaok}}(\lambda)} &\left\{\left(\mathsf{OUT}_\mathcal{A},\ \mathsf{val}_b(\widetilde{\tau})\right) : (\mathsf{OUT}_\mathcal{A}, b) \leftarrow \langle \mathcal{A}_\lambda(\mathsf{st}_\mathcal{M}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})\right\}_{\lambda\in\mathbb{N}} \end{aligned}. \quad (98)$$

By taking $\overline{\varepsilon}(\lambda)$ to be sufficiently small so that $\overline{\varepsilon}(\lambda) + \varepsilon_{\mathsf{wiaok}}(\lambda) \le \varepsilon(\lambda)$ and defining $\mathcal{SE}(1^\lambda, 1^{\varepsilon^{-1}}, \ldots) := \mathcal{SE}'(1^\lambda, 1^{\overline{\varepsilon}^{-1}}, \ldots)$, Eq. (97) and (98) give Lem. 28.

□

B.5 Extractor $\mathcal{K}$ (Proof of Lem. 30)

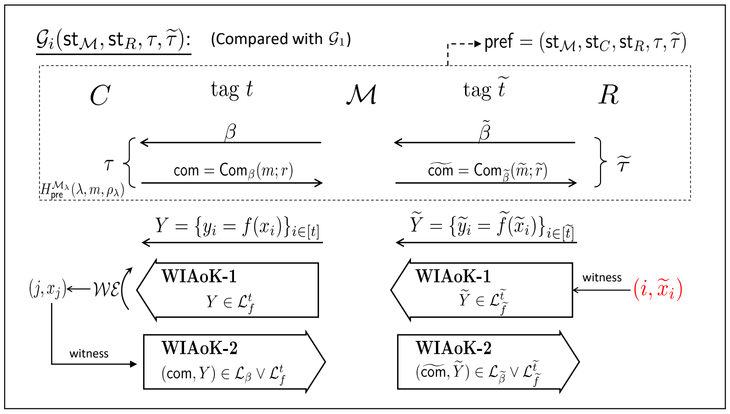
In the following, we fix a noticeable function $\varepsilon(\lambda)$ for which we want to prove Lem. 30. We show that it suffices to set $\varepsilon_{\mathsf{wiaok}}(\lambda) := \frac{t+1}{t^2+6t+3} \cdot \varepsilon'(\lambda) = \frac{t+1}{10t^2(t^2+6t+3)} \cdot \varepsilon(\lambda)$. Since we fix ε and $\varepsilon_{\mathsf{wiaok}}$, we omit the dependence on $\varepsilon_{\mathsf{wiaok}}$ in our notation, i.e., we simply write $p^{\mathsf{Sim}}_{\mathsf{pref}}$ and $\mathcal{G}_1$ to mean $p^{\mathsf{Sim}}_{\mathsf{pref}}[\varepsilon_{\mathsf{wiaok}}]$ and $\mathcal{G}_1[\varepsilon_{\mathsf{wiaok}}]$. Similarly, machines introduced in the following also depend on $\varepsilon_{\mathsf{wiaok}}$, but we do not explicitly write it in our notation. We also omit writing $(1^\lambda, 1^{\varepsilon^{-1}}, \mathcal{A}_\lambda)$ from inputs of those machines for notational simplicity.

Machine $\mathcal{G}_i$ ($i \in [\widetilde{t}]$): (Illustrated in Fig. 13.) Recall that we have already defined the machine $\mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ on Page 81. Now, for $i \in [\widetilde{t}] \setminus \{1\}$, $\mathcal{G}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it uses $(i, \widetilde{x}_i)$ as the witness in the right **WIAoK-1**.

Claim 30. $\forall i \in [\widetilde{t}]$, $\Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] \ge p^{\mathsf{Sim}}_{\mathsf{pref}} - 2\varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda)$.

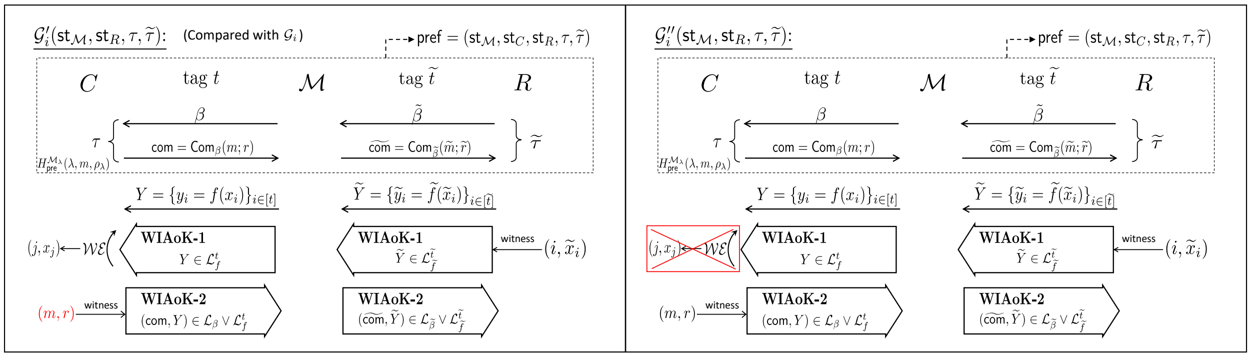
Proof. We define hybrid machines $\mathcal{G}'_i$ and $\mathcal{G}''_i$ as follows.

[38] Remark that $\varepsilon(\lambda)$ and $\frac{\varepsilon'(\lambda)}{\widetilde{t}}$ in Lem. 30 play the roles of $\gamma(\lambda)$ and $\delta(\lambda)$ in Lem. 20, respectively.



$\mathcal{G}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
(Compared with $\mathcal{G}_1$)
$\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\widetilde{\beta}$
τ
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
$\widetilde{\tau}$
$H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$
$Y = \{y_i = f(x_i)\}_{i\in[t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i\in[\widetilde{t}]}$
$(j, x_j) \leftarrow \mathcal{WE}$
WIAoK-1
$Y \in \mathcal{L}^t_f$
WIAoK-1
$\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
witness
$(i, \widetilde{x}_i)$
witness
WIAoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$
WIAoK-2
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$

Fig. 13: Machine $\mathcal{G}_i$ (Difference with $\mathcal{G}_1$ is highlighted in red color)



$\mathcal{G}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
(Compared with $\mathcal{G}_i$)
$\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
$(j, x_j) \leftarrow \mathcal{WE}$
(m, r)
witness
$(i, \widetilde{x}_i)$
$\mathcal{G}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
$\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
$\cancel{(j, x_j) \leftarrow \mathcal{WE}}$
(m, r)
witness
$(i, \widetilde{x}_i)$
WIAoK-1
WIAoK-2

(a) (b)

Fig. 14: Machines $\mathcal{G}'_i$ and $\mathcal{G}''_i$ (Difference is highlighted in red color)

Machine $\mathcal{G}'_i$ ($i \in [\widetilde{t}]$): (Illustrated in Fig. 14a.) Recall that we have already defined the machine $\mathcal{G}'_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ on Page 82. Now, for $i \in [\widetilde{t}]\backslash\{1\}$, $\mathcal{G}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it uses $(i, \widetilde{x}_i)$ as the witness in the right **WIAoK-1**.

Machine $\mathcal{G}''_i$ ($i \in [\widetilde{t}]$): (Illustrated in Fig. 14b) For any prefix pref, $\mathcal{G}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{G}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it honestly runs the left **WIAoK-1** instead of running the witness-extended emulator. In other words, $\mathcal{G}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\langle \mathcal{A}_\lambda(\mathsf{st}_\mathcal{M}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})$ except that it uses $(i, \widetilde{x}_i)$ as the witness in the right **WIAoK-1**. In particular, $\mathcal{G}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ is identical to $\langle \mathcal{A}_\lambda(\mathsf{st}_\mathcal{M}), \mathcal{B}(\mathsf{st}_C, \mathsf{st}_R)\rangle(1^\lambda, \widetilde{\tau})$.

Then, Claim 30 follows from the following sequence of inequalities.

– By the WI property of the left **WIAoK-2** and the definition of $p^{\mathsf{Sim}}_{\mathsf{pref}}$, it holds that:

$$\Pr\big[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}'_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \mathsf{negl}(\lambda).$$

– By the AoK property of the left **WIAoK-1** and the above inequality, it holds that:

$$\Pr\big[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}''_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda).$$

– By the WI property of the right **WIAoK-1** and the above inequality, it holds that:

$$\forall i \in [\widetilde{t}],\ \ \Pr\big[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - \varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda).$$

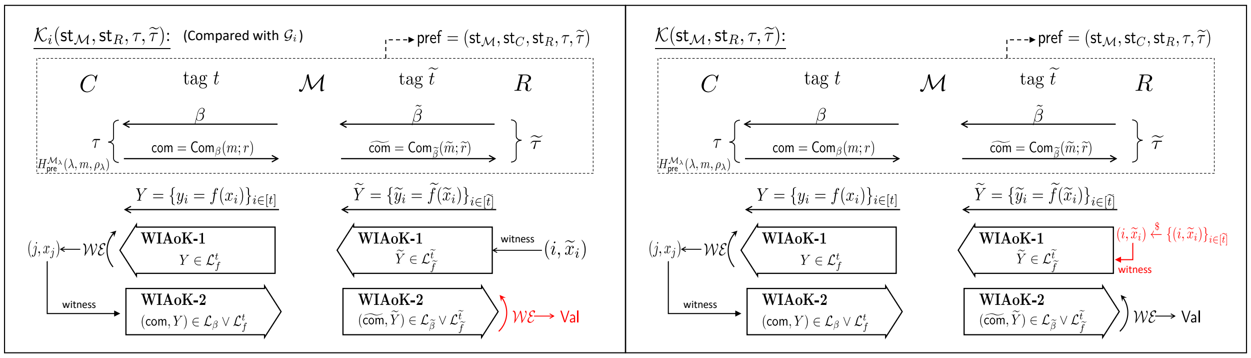
– By the AoK property of the left **WIAoK-1** and the above inequality, it holds that:

$$\forall i \in [\tilde{t}], \;\; \Pr\big[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \tilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - 2\varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda).$$

– By the WI property of the left **WIAoK-2** and the above inequality, it holds that:

$$\forall i \in [\tilde{t}], \;\; \Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \tilde{\tau})] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - 2\varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda).$$

□



$\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \tilde{\tau})$: (Compared with $\mathcal{G}_i$)
$\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \tilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\tilde{t}$
R
β
$\tilde{\beta}$
τ
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\tilde{\beta}}(\tilde{m}; \tilde{r})$
$\tilde{\tau}$
$H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$
$Y = \{y_i = f(x_i)\}_{i \in [t]}$
$\tilde{Y} = \{\tilde{y}_i = \tilde{f}(\tilde{x}_i)\}_{i \in [\tilde{t}]}$
$(j, x_j) \leftarrow \mathcal{WE}$
WIAoK-1
$Y \in \mathcal{L}^t_f$
WIAoK-1
$\tilde{Y} \in \mathcal{L}^{\tilde{t}}_{\tilde{f}}$
witness
$(i, \tilde{x}_i)$
witness
WIAoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$
WIAoK-2
$(\widetilde{\mathsf{com}}, \tilde{Y}) \in \mathcal{L}_{\tilde{\beta}} \vee \mathcal{L}^{\tilde{t}}_{\tilde{f}}$
$\mathcal{WE} \longrightarrow \mathsf{Val}$
$\mathcal{K}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \tilde{\tau})$:
$(i, \tilde{x}_i) \xleftarrow{\$} \{(i, \tilde{x}_i)\}_{i \in [\tilde{t}]}$

(a) (b)

Fig. 15: Machines $\mathcal{K}_i$ and $\mathcal{K}$ (Difference is highlighted in red color)

Machine $\mathcal{K}_i$ ($i \in [\tilde{t}]$): (Illustrated in Fig. 15a.) For a prefix pref, $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \tilde{\tau})$ behaves identically to the $\mathcal{G}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \tilde{\tau})$ depicted in Fig. 13, except for the following difference. Machine $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \tilde{\tau})$ uses the witness-extended emulator (as per Property 2 of Def. 14) with the error parameter $\varepsilon_{\mathsf{wiaok}}(\lambda)$ to finish the right **WIAoK-2**, instead of playing the role of the honest receiver.

$\mathcal{K}_i$'s Output: Let w' denote the third output of the witness-extended emulator (see Property 2 of Def. 14), which is supposed to be the witness used by $\mathcal{M}$ in the right **WIAoK-2** (for the statement $(\widetilde{\mathsf{com}}, \tilde{Y})$ w.r.t. the language $\mathcal{L}_{\tilde{\beta}} \vee \mathcal{L}^{\tilde{t}}_{\tilde{f}}$). Depending on the value of w', we define a value $\mathsf{Val} \in \{0,1\}^{\ell(\lambda)} \cup \{\bot_{\tilde{Y}}, \bot_{\mathsf{invalid}}\}$ as follows:

1. If w' is a valid witness for $(\widetilde{\mathsf{com}}, \tilde{Y}) \in \mathcal{L}_{\tilde{\beta}} \vee \mathcal{L}^{\tilde{t}}_{\tilde{f}}$. Then, there are tow sub-cases:
   (a) w' is a valid witness for $\widetilde{\mathsf{com}} \in \mathcal{L}_{\tilde{\beta}}$. In this case, w' consists of the value $\mathsf{val}(\tilde{\tau})$, i.e., the value committed in $\tilde{\tau} = (\tilde{\beta}, \widetilde{\mathsf{com}})$, and the randomness $\tilde{r}$. We set $\mathsf{Val} := \mathsf{val}(\tilde{\tau})$. Importantly, notice that we do *not* include the randomness $\tilde{r}$ in Val.
   (b) w' is a valid witness for $\tilde{Y} \in \mathcal{L}^{\tilde{t}}_{\tilde{f}}$: In this case, we set $\mathsf{Val} := \bot_{\tilde{Y}}$.
2. Otherwise, set $\mathsf{Val} := \bot_{\mathsf{invalid}}$.

The output of $\mathcal{K}_i$ is defined to be the above Val. Notice that this is in contrast to all previous machines, for which the output is defined to be the main-in-the-middle's output and the honest receiver's decision bit. We emphasize that such a Val satisfies the syntax requirement in Property 1 of Lem. 30. In particular, $\mathsf{Val} = \mathsf{val}(\tilde{\tau})$ *whenever* $\mathsf{Val} \neq \bot$.[39]

[39] Note that here we defined two types of abortion: $\bot_{\tilde{Y}}$ and $\bot_{\mathsf{invalid}}$, while Property 1 of Lem. 30 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: $\bot = \bot_{\tilde{Y}}$ *and* $\bot = \bot_{\mathsf{invalid}}$ (i.e., $\mathsf{Val} \neq \bot \Leftrightarrow (\mathsf{Val} \neq \bot_{\tilde{Y}} \wedge \mathsf{Val} \neq \bot_{\mathsf{invalid}})$).

Claim 31. $\forall i \in [\widetilde{t}]$, $\Pr[\mathsf{Val} \neq \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - 3\varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda)$.

Proof. This claim follows from Claim 30 and the AoK property of the right **WIAoK-2**. □

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

Extractor $\mathcal{K}$: (Illustrated in Fig. 15b.) For a prefix pref, $\mathcal{K}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ samples uniformly at random an index $i \xleftarrow{\$} [\widetilde{t}]$, executes $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$, and outputs whatever $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ outputs.

Next, we show that the extractor $\mathcal{K}$ satisfies the requirements in Lem. 30.

Running Time of $\mathcal{K}$. Observe that for each $i \in [\widetilde{t}]$, $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ differs from the real man-in-the-middle execution only in the following places:

- $(i, \widetilde{x}_i)$ is used in the right **WIAoK-1**;
- the witness-extended emulator is used in the right **WIAoK-2** and the left **WIAoK-1**.

Since the witness-extended emulator (as per Property 2 of Def. 14) runs in QPT, so does $\mathcal{K}_i$. Thus, $\mathcal{K}$ is QPT.

Proving Property 1 of Lem. 30. It is straightforward to see that the $\mathcal{K}$ defined above satisfies the syntax requirement in Lem. 30. In particular, we have $\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau})$ whenever $\mathsf{Val}_{\mathcal{K}} \neq \bot$, because this is true for each $\mathcal{K}_i$ by definition (see the paragraph for "$\mathcal{K}_i$'s Output" on Page 86).

Proving Property 2 of Lem. 30. First, recall that Property 2 requires that for any pref in the support of $H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$, if $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$, then it holds that

$$\Pr[\mathsf{Val}_{\mathcal{K}} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val}_{\mathcal{K}} \leftarrow \mathcal{K}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq \frac{\varepsilon'(\lambda)}{\widetilde{t}}. \tag{99}$$

Also recall that $\mathcal{K}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ is defined to execute the machine $\mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ with i uniformly sampled from $[\widetilde{t}]$. Therefore, Inequality (99) can be reduced to the following Lem. 31. We will prove Lem. 31 in Appx. B.6, which will eventually finish the current proof of Lem. 30.

Lemma 31. *Let $\varepsilon(\lambda) = \frac{1}{\mathsf{poly}(\lambda)}$ and $\varepsilon'(\lambda) = \frac{\varepsilon(\lambda)}{t^2}$. For any $\mathsf{pref} = (\mathsf{st}_{\mathcal{M}}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$, if $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$, then there exists an $i \in [\widetilde{t}]$ such that*

$$\Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq \varepsilon'(\lambda).$$

B.6 Proof of Lem. 31

Notation. We highly recommend reviewing the "$\mathcal{K}_i$'s Output" part on Page 86 (in particular, the meanings of Val, $\bot_{\widetilde{Y}}$, and $\bot_{\mathsf{invalid}}$) before starting reading this subsection. Recall that $\mathcal{K}_i$'s output Val is determined by the w' output by the witness-extended emulator of the right **WIAoK-2**. In this subsection, we will need to refer to this w', although it is not included as a part of $\mathcal{K}_i$'s output. Particularly, we will make use of the following notation: whenever we write an expression of the form

$$\Pr\big[\text{Some Event } E_{w'} \text{ about } w' : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big],$$

it should be understood as the probability of $E_{w'}$, where w' refers to the w' generated during the random procedure $\mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$, over which the probability is taken.

Using these notations, we can partition the event $\mathsf{Val} = \bot_{\widetilde{Y}}$ as the following *mutually exclusive* events: $w' = (1, \widetilde{x}_1)$, or, …, or $w' = (\widetilde{t}, \widetilde{x}_{\widetilde{t}})$, where $\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)$ for each $i \in [\widetilde{t}]$. Formally, we express this relation by

$$\Pr\big[\mathsf{Val} = \bot_{\widetilde{Y}} : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] = \sum_{i=1}^{\widetilde{t}} \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \tag{100}$$

With the above notations, we prove Lem. 31 in the following.

Proof of Lem. 31. We assume for contradiction that for some pref satisfying $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$, it holds that

$$\forall i \in [\widetilde{t}],\ \ \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau}) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})] < \varepsilon'(\lambda). \tag{101}$$

Claim 32. *Under the assumption in Inequality (101), it holds that*

$$\forall i \in [\widetilde{t}],\ \ \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} - 3\varepsilon_{\mathsf{wiaok}}(\lambda) - \varepsilon'(\lambda) - \mathsf{negl}(\lambda).$$

Proof. In this proof, all the probabilities are taken over the random procedure $\mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$.

First, notice that

$$\begin{aligned}\forall i \in [\widetilde{t}],\ \ \Pr[\mathsf{Val} \neq \bot_{\mathsf{invlid}}] &= \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] + \Pr\big[\mathsf{Val} = \bot_{\widetilde{Y}}\big] \\ \text{(by Eq. (100))} \ \ &= \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] + \Pr\big[w' = (i, \widetilde{x}_i)\big] + \sum_{j \in [\widetilde{t}] \setminus \{i\}} \Pr\big[w' = (j, \widetilde{x}_j)\big]. \end{aligned} \tag{102}$$

Then, the following holds:

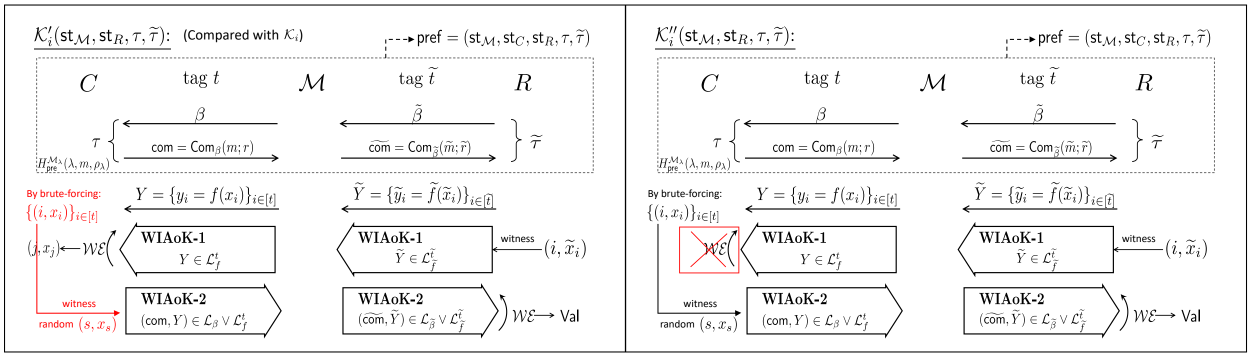
$$\begin{aligned}\forall i \in [\widetilde{t}],\ \ \Pr[w' = (i, \widetilde{x}_i)] &= \Pr[\mathsf{Val} \neq \bot_{\mathsf{invalid}}] - \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] - \Big(\sum_{j \in [\widetilde{t}] \setminus \{i\}} \Pr[w' = (j, \widetilde{x}_j)]\Big) &(103)\\ &\geq p^{\mathsf{Sim}}_{\mathsf{pref}} - 3\varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda) - \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] - \Big(\sum_{j \in [\widetilde{t}] \setminus \{i\}} \Pr[w' = (j, \widetilde{x}_j)]\Big) &(104)\\ &\geq p^{\mathsf{Sim}}_{\mathsf{pref}} - 3\varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda) - \varepsilon'(\lambda) - \Big(\sum_{j \in [\widetilde{t}] \setminus \{i\}} \Pr[w' = (j, \widetilde{x}_j)]\Big), &(105)\end{aligned}$$

where Eq. (103) follows from Eq. (102), Inequality (104) follows from Claim 31, and Inequality (105) follows from Inequality (101).

Now, to prove Claim 32, it suffices to show that

$$\forall i \in [\widetilde{t}],\ \forall j \in [\widetilde{t}] \setminus \{i\},\ \ \Pr\big[w' = (j, \widetilde{x}_j) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] = \mathsf{negl}(\lambda). \tag{106}$$

This can be reduced via standard techniques to the (post-quantum) one-wayness of the OWF $\widetilde{f}$ in Step 3 of the right execution. In more details, we assume for contradiction that there exist $i^*, j^* \in [\widetilde{t}]$ such that $i^* \neq j^*$ and that the probability $\Pr\big[w' = (j^*, \widetilde{x}_{j^*}) : \mathsf{Val} \leftarrow \mathcal{K}_{i^*}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big]$ is non-negligible, where, by definition, $\widetilde{x}_{j^*}$ is the preimage of $\widetilde{y}_{j^*}$ under the right OWF $\widetilde{f}$. Then, we can build a QPT adversary $\mathcal{A}_{\text{OWF}}$ breaking one-wayness in the following way: $\mathcal{A}_{\text{OWF}}$ obtains the challenge y^* from the external one-wayness challenger; it then runs the machine $\mathcal{K}_{i^*}(\mathsf{pref})$ internally, for which $\mathcal{A}_{\text{OWF}}$ uses y^* in place


$\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
(Compared with $\mathcal{K}_i$)
$\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\beta}$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
τ
$\widetilde{\tau}$
$H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$
By brute-forcing:
$\{(i, x_i)\}_{i \in [t]}$
$Y = \{y_i = f(x_i)\}_{i \in [t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i \in [\widetilde{t}]}$
$(j, x_j) \leftarrow \mathcal{WE}$
WIAoK-1
$Y \in \mathcal{L}^t_f$
WIAoK-1
$\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
witness
$(i, \widetilde{x}_i)$
witness
random (s, x_s)
WIAoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$
WIAoK-2
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
$\mathcal{WE} \rightarrow$ Val
$\mathcal{K}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
$\mathsf{pref} = (\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\beta}$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
τ
$\widetilde{\tau}$
$H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$
By brute-forcing:
$\{(i, x_i)\}_{i \in [t]}$
$Y = \{y_i = f(x_i)\}_{i \in [t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i \in [\widetilde{t}]}$
$\mathcal{WE}$
WIAoK-1
$Y \in \mathcal{L}^t_f$
WIAoK-1
$\widetilde{Y} \in \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
witness
$(i, \widetilde{x}_i)$
witness
random (s, x_s)
WIAoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}^t_f$
WIAoK-2
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}^{\widetilde{t}}_{\widetilde{f}}$
$\mathcal{WE} \rightarrow$ Val

(a) (b)

Fig. 16: Machines $\mathcal{K}'_i$ and $\mathcal{K}''_i$ (Difference is highlighted in red color)

of $\widetilde{y}_{j^*}$ when executing Step 3 in the right. Note that the internal execution of $\mathcal{K}_{i^*}(\mathsf{pref})$ is identically to the real execution of $\mathcal{K}_{i^*}$, thus the extracted $w' = (j^*, \widetilde{x}_{j^*})$ must satisfy $\widetilde{f}(\widetilde{x}_{j^*}) = \widetilde{y}^*_j\ (= y^*)$ with non-negligible probability, breaking one-wayness.

This finishes the proof of Claim 32.

□

Machine $\mathcal{K}'_i$ $(i \in [\widetilde{t}])$: (Illustrated in Fig. 16a.) For a prefix pref, $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ proceeds as follows:

1. It first finishes Step 3 of the man-in-the-middle execution in the same way as $\mathcal{K}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$. In particular, it will see in the left execution the values $Y = (y_1, \ldots, y_t)$ sent from $\mathcal{M}$;
2. It then recovers $(x_1, \ldots, x_t)$ by brute-force: Namely, for each $i \in [t]$, it inverts the OWF f to find x_i s.t. $f(x_i) = y_i$. It is possible that there exist some "bad" y_i's that are not in the range of f. For such bad i's, it sets $x_i = \bot$. If all the x_i's are bad, $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ halts and outputs Fail;
3. If this step is reached, we know that $(x_1, \ldots, x_t)$ cannot be all-$\bot$. $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ then picks an (s, x_s) uniformly at random from the good (i.e. non-$\bot$) x_i's.
4. Then, $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ continues to finish the execution in the same way as $\mathcal{K}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$, except that it uses (s, x_s) as the witness when executing the left **WIAoK-2**.

It is worth noting that the (j, x_j) extracted by the witness-extended emulator for the left **WIAoK-1** (inherited from $\mathcal{K}_i(\mathsf{pref})$) is not used any more by $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$.

It is easy to see that if the (s, x_s) picked by $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ is equal to the (j, x_j) extracted in $\mathcal{K}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ from its left **WIAoK-1**, then $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ and $\mathcal{K}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ are identical.[40] Since $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ samples (s, x_s) uniformly from all the good (i, x_i)'s, it must hold with probability at least $1/t$ that $(s, x_s) = (j, x_j)$. Therefore, the following must hold:

$$\forall i \in [\widetilde{t}],\ \ \Pr[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] \geq \frac{1}{t} \cdot \Pr[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]. \tag{107}$$

Machine $\mathcal{K}''_i$ $(i \in [\widetilde{t}])$: (Illustrated in Fig. 16b.) For a prefix pref, $\mathcal{K}''_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically to $\mathcal{K}'_i(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that it plays as the honest committer in the left **WIAoK-1**, instead of running the witness-extended emulator. Recall that starting from $\mathcal{K}'_i$, the witness (j, x_j)

[40] Similar to Rmk. 11, this is the only place we make use of the injectivity of f.

extracted by the witness-extended emulator from the left **WIAoK-1** is not used any more. Thus, machine $\mathcal{K}''_i$ does not need to perform this witness-extended emulation.

By the AoK property of the left **WIAoK-1**, it holds that

$$\forall i \in [\widetilde{t}], \ \Big|\Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] - \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}'_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big]\Big| \le \varepsilon_{\mathsf{wiaok}}(\lambda) + \mathsf{negl}(\lambda). \tag{108}$$

Next, by the (non-uniform) WI property of the right **WIAoK-1**, it holds that

$$\forall i \in [\widetilde{t}], \ \Big|\Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_i(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] - \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big]\Big| \le \mathsf{negl}(\lambda). \tag{109}$$

Remark 21 (Power of Non-Uniform Reductions). Note that we can rely on the AoK and WI properties even though $\mathcal{K}'_i$ and $\mathcal{K}''_i$ perform brute-force to recover $(x_1, ..., x_t)$. This is because the brute-forcing step is done before **WIAoK-1** or **WIAoK-2** starts; Thus, $(x_1, ..., x_t)$ can be treated as a non-uniform advice in the reductions. This non-uniform type of argument will be used again in this section later.

Then, we have the following claim:

Claim 33. *It holds that*

$$\forall i \in [\widetilde{t}], \ \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \ge \frac{1}{t} \cdot \big(p^{\mathsf{Sim}}_{\mathsf{pref}} - (t+3)\varepsilon_{\mathsf{wiaok}}(\lambda) - \varepsilon'(\lambda)\big) - \mathsf{negl}(\lambda).$$

Proof. This claim follows from Claim 32 and Inequalities (35), (36) and (107). Formally, (in the following, we omit the input $(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ to $\mathcal{K}_i$, $\mathcal{K}'_i$, and $\mathcal{K}''_i$)

$$\forall i \in [\widetilde{t}], \ \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_1\big] \ge \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}''_i\big] - \mathsf{negl}(\lambda) \tag{110}$$
$$\ge \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}'_i\big] - \varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda) \tag{111}$$
$$\ge \frac{1}{t} \cdot \Pr\big[w' = (i, \widetilde{x}_i) : \mathsf{Val} \leftarrow \mathcal{K}_i\big] - \varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda) \tag{112}$$
$$\ge \frac{1}{t} \cdot \big(p^{\mathsf{Sim}}_{\mathsf{pref}} - 3\varepsilon_{\mathsf{wiaok}}(\lambda) - \varepsilon'(\lambda)\big) - \varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda), \tag{113}$$

where Inequality (110) follows from Inequality (109), Inequality (111) follows from Inequality (108), Inequality (112) follows from Inequality (107), and Inequality (113) follows from Claim 32.

□

Now, we show the last claim which, together with Claim 33, will lead to the desired contradiction.

Claim 34. *It holds that*

$$\Pr\big[\mathsf{Val} \ne \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}''_1(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})\big] \le p^{\mathsf{Sim}}_{\mathsf{pref}} + 2\varepsilon_{\mathsf{wiaok}}(\lambda) + \mathsf{negl}(\lambda).$$

Deriving the Contradiction. Before proving Claim 34, we first show why Claims 10 and 33 are contradictory. In the following, all the probabilities are taken over $\mathsf{Val} \leftarrow \mathcal{K}_1''(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$:

$$\begin{aligned}
\Pr[\mathsf{Val} \neq \bot_{\mathsf{invalid}}] &= \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] + \Pr\big[\mathsf{Val} = \bot_{\widetilde{Y}}\big] \\
&= \Pr[\mathsf{Val} = \mathsf{val}(\widetilde{\tau})] + \sum_{i=1}^{\widetilde{t}} \Pr\big[w' = (i, \widetilde{x}_i)\big] && (114)\\
&\geq \sum_{i=1}^{\widetilde{t}} \Pr\big[w' = (i, \widetilde{x}_i)\big] \\
&\geq \widetilde{t} \cdot \frac{1}{t} \cdot \big(p_{\mathsf{pref}} - (t+3)\varepsilon_{\mathsf{wiaok}}(\lambda) - \varepsilon'(\lambda)\big) - \mathsf{negl}(\lambda) && (115)\\
&\geq (1 + \frac{1}{t}) \cdot (p_{\mathsf{pref}} - (t+3)\varepsilon_{\mathsf{wiaok}}(\lambda) - \varepsilon'(\lambda)) - \mathsf{negl}(\lambda) && (116)\\
&= (1 + \frac{1}{t}) \cdot \left(p_{\mathsf{pref}} - \frac{t^2+6t+3}{t+1} \cdot \varepsilon_{\mathsf{wiaok}}(\lambda) - \varepsilon'(\lambda)\right) + 2\varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda) \\
&= (1 + \frac{1}{t}) \cdot \left(p_{\mathsf{pref}} - 2\varepsilon'(\lambda)\right) + 2\varepsilon_{\mathsf{wiaok}}(\lambda) - \mathsf{negl}(\lambda) && (117)\\
&= p^{\mathsf{Sim}}_{\mathsf{pref}} + 2\varepsilon_{\mathsf{wiaok}}(\lambda) + \left(\frac{p^{\mathsf{Sim}}_{\mathsf{pref}}}{t} - 2\varepsilon'(\lambda) - \frac{2\varepsilon'(\lambda)}{t}\right) - \mathsf{negl}(\lambda) \\
&\geq p^{\mathsf{Sim}}_{\mathsf{pref}} + 2\varepsilon_{\mathsf{wiaok}}(\lambda) + \frac{5t^2 - t - 1}{5t^3} \cdot \varepsilon(\lambda) - \mathsf{negl}(\lambda)) && (118)
\end{aligned}$$

where Eq. (114) follows from Eq. (100), Inequality (115) follows from Claim 33, Inequality (116) follows from the assumption that $\widetilde{t} \geq t+1$, Eq. (117) follows from our parameter setting $\varepsilon_{\mathsf{wiaok}}(\lambda) = \frac{t+1}{t^2+6t+3} \cdot \varepsilon'(\lambda)$, and Inequality (118) follows from the assumption that $p^{\mathsf{Sim}}_{\mathsf{pref}} \geq \varepsilon(\lambda)$ 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 (118) can be written as:

$$\Pr\big[\mathsf{Val} \neq \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}_1''(\mathsf{pref})\big] \geq p^{\mathsf{Sim}}_{\mathsf{pref}} + \frac{1}{\mathsf{poly}(\lambda)} - \mathsf{negl}(\lambda),$$

which contradicts Claim 34.

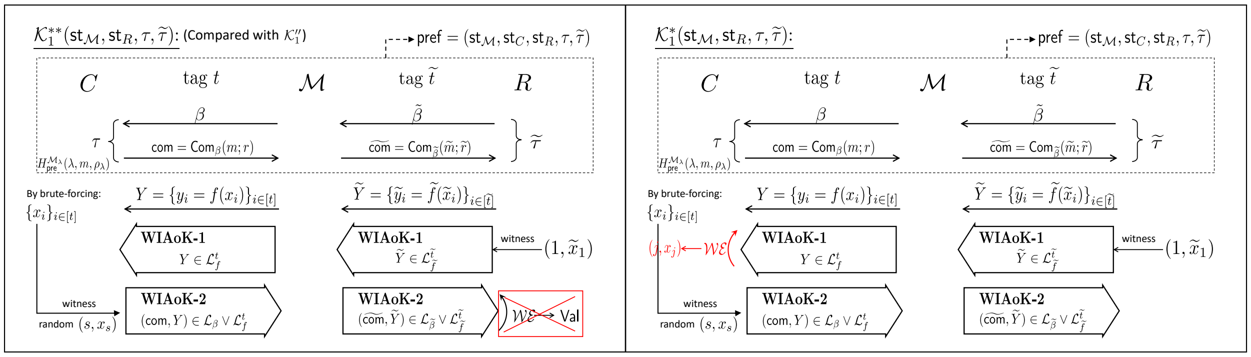
This eventually finishes the proof of Lem. 31 (modulo the proof of Claim 34, which we show in Appx. B.7).

B.7 Proof of Claim 34

We first define two extra machines $\mathcal{K}_1^{**}$ and $\mathcal{K}_1^{*}$.

Machine $\mathcal{K}_1^{**}$: (Illustrated in Fig. 17a.) For a prefix pref, $\mathcal{K}_1^{**}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically as $\mathcal{K}_1''(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except that $\mathcal{K}_1^{**}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ finishes the right **WIAoK-2** using the honest receiver's algorithm, instead of using the witness-extended emulator.

$\mathcal{K}_1^{**}$'s Output. We define the output of $\mathcal{K}_1^{**}(\mathsf{st}_{\mathcal{M}}, \mathsf{st}_R, \tau, \widetilde{\tau})$ to be the honest receiver's decision $b \in \{\top, \bot\}$. This is in contrast to previous hybrids $\mathcal{K}_i'$, $\mathcal{K}_i''$, and $\mathcal{K}_i$, whose output is defined to be the value Val that depends on the value w' extracted by the $\mathcal{SE}$ for the right **WIAoK-2**.



$\mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$: (Compared with $\mathcal{K}_1''$)
pref = $(\mathsf{st}_\mathcal{M}, \mathsf{st}_C, \mathsf{st}_R, \tau, \widetilde{\tau})$
C
tag t
$\mathcal{M}$
tag $\widetilde{t}$
R
β
$\widetilde{\beta}$
$\mathsf{com} = \mathsf{Com}_\beta(m; r)$
$\widetilde{\mathsf{com}} = \mathsf{Com}_{\widetilde{\beta}}(\widetilde{m}; \widetilde{r})$
$H^{\mathcal{M}_\lambda}_{\mathsf{pre}}(\lambda, m, \rho_\lambda)$
By brute-forcing: $\{x_i\}_{i \in [t]}$
$Y = \{y_i = f(x_i)\}_{i \in [t]}$
$\widetilde{Y} = \{\widetilde{y}_i = \widetilde{f}(\widetilde{x}_i)\}_{i \in [\widetilde{t}]}$
WIAoK-1
$Y \in \mathcal{L}_f^t$
$\widetilde{Y} \in \mathcal{L}_{\widetilde{f}}^{\widetilde{t}}$
witness
$(1, \widetilde{x}_1)$
WIAoK-2
$(\mathsf{com}, Y) \in \mathcal{L}_\beta \vee \mathcal{L}_f^t$
$(\widetilde{\mathsf{com}}, \widetilde{Y}) \in \mathcal{L}_{\widetilde{\beta}} \vee \mathcal{L}_{\widetilde{f}}^{\widetilde{t}}$
random (s, x_s)
$\mathcal{WE}$ → Val
$\mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$:
$(j, x_j) \leftarrow \mathcal{WE}$
(a)
(b)

Fig. 17: Machines $\mathcal{K}_1^{**}$ and $\mathcal{K}_1^{*}$ (Difference is highlighted in red color)

By the AoK property of the right **WIAoK-2**, it holds that

$$\left|\Pr\left[\mathsf{Val} \neq \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}_1''(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\right] - \Pr[b = \top : b \leftarrow \mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]\right| \leq \varepsilon_{\mathsf{wiaok}}(\lambda) + \mathsf{negl}(\lambda). \quad (119)$$

Machine $\mathcal{K}_1^*$: (Illustrated in Fig. 17b.) For a prefix pref, $\mathcal{K}_1^*(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ behaves identically as $\mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ except the following difference: $\mathcal{K}_1^*(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})$ uses the witness-extended emulator with the error parameter $\varepsilon_{\mathsf{wiaok}}(\lambda)$ to extract a witness (j, x_j) from the left **WIAoK-1**, and if x_j is not a valid preimage for y_j, $\mathcal{K}_1^*$ aborts.

By the AoK property of the left **WIAoK-1**, it holds that

$$\left|\Pr[b = \top : b \leftarrow \mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] - \Pr[b = \top : b \leftarrow \mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]\right| \leq \varepsilon_{\mathsf{wiaok}}(\lambda) + \mathsf{negl}(\lambda). \quad (120)$$

Compare $\mathcal{K}_1^*$ with $\mathcal{G}_1$. Now, let us compare $\mathcal{K}_1^*$ with the $\mathcal{G}_1$ depicted in Fig. 13. They only differ in the witness used in the left **WIAoK-2** (and that $\mathcal{G}_1$ does not need to perform brute-forcing for Y, as it does not use those preimages). Therefore, by the (non-uniform) WI property of the left **WIAoK-2**, it holds that

$$\left|\Pr[b = \top : b \leftarrow \mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] - \Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]\right| \leq \mathsf{negl}(\lambda).$$

Also, recall (from Def. 16) that $\Pr[b = \top : (\mathsf{OUT}, b) \leftarrow \mathcal{G}_1(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})]$ is exactly the definition of $p^{\mathsf{Sim}}_{\mathsf{pref}}$. Thus, the above implies:

$$\left|\Pr[b = \top : b \leftarrow \mathcal{K}_1^{*}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] - p^{\mathsf{Sim}}_{\mathsf{pref}}\right| \leq \mathsf{negl}(\lambda). \quad (121)$$

Therefore, the following holds:

$$\begin{aligned}
\Pr\left[\mathsf{Val} \neq \bot_{\mathsf{invalid}} : \mathsf{Val} \leftarrow \mathcal{K}_1''(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})\right] &\leq \Pr[b = \top : b \leftarrow \mathcal{K}_1^{**}(\mathsf{st}_\mathcal{M}, \mathsf{st}_R, \tau, \widetilde{\tau})] + \varepsilon_{\mathsf{wiaok}}(\lambda) + \mathsf{negl}(\lambda) && (122)\\
&\leq \Pr[b = \top : b \leftarrow \mathcal{K}_1^{*}(\mathsf{pref})] + 2\varepsilon_{\mathsf{wiaok}}(\lambda) + \mathsf{negl}(\lambda) && (123)\\
&\leq p^{\mathsf{Sim}}_{\mathsf{pref}} + 2\varepsilon_{\mathsf{wiaok}}(\lambda) + \mathsf{negl}(\lambda), && (124)
\end{aligned}$$

where Inequality (122) follows from Inequality (119), Inequality (123) follows from Inequality (120), and Inequality (124) follows from Inequality (121).

This finishes the proof of Claim 34.