GUC-Secure Commitments via Random Oracles: New Impossibility and Feasibility[*]

Zhelei Zhou[a,b], Bingsheng Zhang[†] [‡a,b], Hong-Sheng Zhou [†] [§c], and Kui Ren[a,b]

[a]Zhejiang University
[b]ZJU-Hangzhou Global Scientific and Technological Innovation Center
[c]Virginia Commonwealth University

August 30, 2022

Abstract

In the UC framework, a protocol must be subroutine respecting; therefore, shared trusted setup might cause security issues. To address this drawback, Generalized UC (GUC) framework is introduced by Canetti *et al.* (TCC 2007). In this work, we investigate the impossibility and feasibility of GUC-secure commitments with global random oracles (GRO) as trusted setup. In particular, we show it is impossible to have a 2-round (1 round for the committing phase and 1 round for the opening phase) GUC-secure commitment in the global observable RO model by Canetti *et al.* (CCS 2014). We then give a new round-optimal GUC-secure commitment that uses only MiniCrypt assumptions (i.e. the existence of one-way functions) in the global observable RO model. In addition, we also examine the complete picture on round complexity of the GUC-secure commitments in various global RO models.

[*]This is the full version of the paper accepted by Asiacrypt 2022.
[†]Corresponding authors: bingsheng@zju.edu.cn; hszhou@vcu.edu.
[‡]Work supported by the National Key R&D Program of China (No. 2021YFB3101601), the National Natural Science Foundation of China (Grant No. 62072401), and "Open Project Program of Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province". This project is also supported by Input Output (iohk.io).
[§]Work supported in part by NSF grant CNS-1801470, a Google Faculty Research Award and a research gift from Ergo Platform.

Contents

1 Introduction

Secure multi-party computation (MPC) [Yao82, GMW87] allows n mutually distrustful players $(P_1, \ldots, P_n)$ to securely evaluate any efficiently computable function f of their private inputs $(x_1, \ldots, x_n)$. Since its introduction in the early 1980s, MPC has been extensively studied and has become one of the cornerstone of modern cryptography. MPC security properties are formalized using the classic "simulation-paradigm" [GMR89, GMW87]: roughly speaking, the idea is to require that any adversarial attacker $\mathcal{A}$ in the *real world* execution of the protocol, can be emulated by a so-called "simulator" $\mathcal{S}$ in an *ideal world* execution, where the player provide their inputs to a trusted third party who computes f for them and relays back the answer to them.

From UC to GUC. To enable protocol design and analysis in the complex network environments, Canetti proposed the Universal Composibility (UC) framework [Can01]; here, the notion of indistinguishability between the real and the ideal world is replaced by a notion of "interactive indistinguishability": more specifically, an interactive *environment*, which may communicate with both the honest players and the corrupted ones, should not be able to distinguish whether it is participating in the real execution or the ideal one. UC security guarantees security of the MPC protocol even under *concurrent executions* of the protocol, and even that other *arbitrary* protocols that may be running on the network and not adversarially affected by the MPC protocol—roughly speaking, the environment represents these other network protocols. Additionally, this notion is closed under composition, enabling modular analysis of protocols.

However, protocols in the UC framework must be *subroutine respecting* in the sense that the protocol state cannot be shared across the protocol sessions. To address this issue, later Canetti, Dodis, Pass and Walfish proposed the Generalized Universal Composibility (GUC) framework [CDPW07], and interesting and efficient protocols have been designed and analyzed [DSW08, CJS14, MRS17, CDG$^+$18, CSW20].

Random Oracles as a global setup: $\mathcal{G}_{\mathsf{sRO}}$, $\mathcal{G}_{\mathsf{oRO}}$, $\mathcal{G}_{\mathsf{pRO}}$, and $\mathcal{G}_{\mathsf{poRO}}$. It has been shown [CF01, CDPW07] that, to achieve secure multi-party computation for any non-trivial functionality in the UC and the GUC framework, certain trusted setups (e.g., CRS, PKI, etc) are required. Idealized setups such as Random Oracles can also be used for designing UC-secure [HM04] and GUC-secure multi-party computation protocols [CJS14, CDG$^+$18].

Random oracle (RO) model [BR93] is a popular idealized model that enables highly efficient cryptographic protocols. In spite of its known inability to provide provable guarantees when RO is instantiated with a real-world hash function [CGH98], RO is still a promising setup without known real-world attacks. In fact, RO draws increasing attention along with recent advancement of the blockchain technology. It is generally viewed as a *transparent* setup that can be easily deployed with no reliance on any trusted party in the blockchain and/or distributed system setting. Many RO-based non-interactive ZK systems, e.g., zk-STARK [BBHR18] and Fractal [COS20], are developed and deployed in real application scenarios. Note that, those RO-based protocols can achieve post-quantum security.

A natural formulation of global RO denoted as $\mathcal{G}_{\mathsf{sRO}}$ has been defined in [CDPW07]: it is accessible to all parties both in the ideal world and the real world, but provides neither "observability" nor "programmability". We here highlight that, it has proven that it is impossible to achieve GUC-secure commitment in the $\mathcal{G}_{\mathsf{sRO}}$ model [CDPW07]. Later, Canetti, Jain, and Scafuro [CJS14] proposed a strengthened version of global RO denoted as $\mathcal{G}_{\mathsf{oRO}}$, which allows the simulator to "observe" the queries made by the malicious parties, and GUC-secure commitment *can* be constructed in the $\mathcal{G}_{\mathsf{oRO}}$ model. Camenisch *et al.* [CDG$^+$18] further strengthened the $\mathcal{G}_{\mathsf{sRO}}$ from a different direction, they designed a mechanism that allows the simulator to "program" the global RO without being detected, and introduced another strengthened version of global RO denoted as $\mathcal{G}_{\mathsf{pRO}}$. On top of both $\mathcal{G}_{\mathsf{oRO}}$ and $\mathcal{G}_{\mathsf{pRO}}$, Camenisch *et al.* [CDG$^+$18] then introduced an even stronger variant of global RO called $\mathcal{G}_{\mathsf{poRO}}$, and construct a round-optimal GUC-secure commitment in the $\mathcal{G}_{\mathsf{poRO}}$ model [CDG$^+$18]. See the relation of these global RO models in Figure 4.

Using weaker assumptions and our questions. We study the round complexity of GUC-secure commitment using a global RO. It is clear that protocols using a *less idealized* setup and *weaker* computational assumptions will enable better confidence on the security guarantee. Note that, round-optimal GUC secure commitment can be constructed based on a strong RO setup $\mathcal{G}_{\mathsf{poRO}}$ [CDG$^+$18]. On the other hand, in [CDPW07], it has proven that GUC-secure commitment in the $\mathcal{G}_{\mathsf{sRO}}$ model, is impossible. Between the two extremes, in [CJS14], it has shown that GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model, is feasible, but relying on strong computational assumptions in CryptoMania. A natural research question we ask here is:

What is the lower bounds of the round complexity for GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model?

If there exists such a lower bound on the round complexity for GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model, our next step is to provide a round optimal construction. We ask:

If there exists such a lower bound, is that possible to construct round-optimal GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model, using only MiniCrypt assumption?

1.1 Our results

We give affirmative answers to the above research questions. We now summarize our research findings.

A new impossibility result in the $\mathcal{G}_{\mathsf{oRO}}$ model. In this work, we show that 2-round[1] (1 round for the committing phase and 1 round for the opening phase) GUC-secure commitment does not exist in the $\mathcal{G}_{\mathsf{oRO}}$ model (cf. Section 3).

We prove this result by contradiction, and our main observation is as follows. Suppose such a 2-round GUC-secure commitment exists. First, it is easy to see that if the committing phase only takes one round, then there is only one message sent from the committer to the receiver; that is, the receiver does not send any message to the committer. Analogously, the receiver is also "silent" in the 1-round opening phase. Therefore, the potentially corrupted receiver can delay all its $\mathcal{G}_{\mathsf{oRO}}$ queries until it receives the opening message from the committer.

Let us consider the case where the receiver is corrupted. During the simulation, the simulated committer needs to generate the commitment message without the knowledge of the plaintext, and it later needs to generate the opening message for any given input (a.k.a. the plaintext). As discussed before, the corrupted receiver can choose not to query the $\mathcal{G}_{\mathsf{oRO}}$ oracle until the simulator has equivocated the commitment. Hence, the simulator cannot obtain any illegitimate queries from $\mathcal{G}_{\mathsf{oRO}}$ for this corrupted receiver to facilitate this equivocation. Now, observe that this simulator has no extra power over a normal party; in particular, any committer can invoke such a simulator (algorithm) to violate the binding property of the commitment.

In the actual proof of our impossibility result, we let the corrupted committer to internally run the simulator algorithm to generate the commitment message, providing an empty list for the $\mathcal{G}_{\mathsf{oRO}}$ illegitimate queries. Obviously, given this commitment message, the receiver/simulator cannot extract its plaintext; with very high probability the simulation would fail.

A new round-optimal commitment using $\mathcal{G}_{\mathsf{oRO}}$. With respect to our impossibility result, a round-optimal commitment should takes at least 3 rounds. In this work, we show how to construct a round-optimal (2 rounds for the committing phase and 1 round for the opening phase) GUC-secure commitment only using MiniCrypt assumptions in the $\mathcal{G}_{\mathsf{oRO}}$ model (cf. Section 4).

A general framework. A typical GUC-secure commitment requires both extractability and equivocality. The $\mathcal{G}_{\mathsf{oRO}}$ model can directly provide the simulator extractability; therefore, the challenge is to design an equivocation mechanism with round efficiency. A natural approach is to utilize a (property-based) perfect-hiding non-interactive equivocal commitment: (i) in the first round, the receiver picks the commitment key and sends it to the committer; and (ii) in the second round, the committer uses the equivocal commitment scheme to commit the message. Hereby, the following two questions need to be resolved:

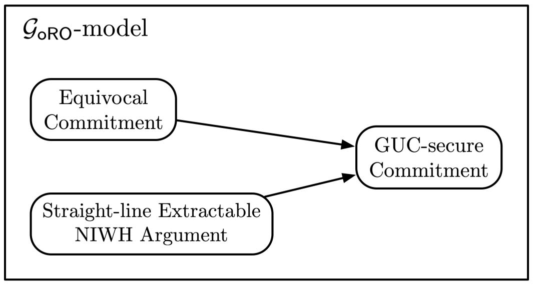
- How shall we instantiate such a perfect-hiding non-interactive equivocal commitment?
- How can the simulator obtain the equivocation trapdoor?

In [CJS14] and [MRS17], the Pedersen commitment is used as a candidate of the equivocal commitment. It is well-known, the security of Pedersen commitment is based on the DL assumption, which lives in the CryptoMania world. In the next paragraph, we will show how to construct a candidate of the equivocal commitment only using MiniCrypt assumptions, i.e. the existence of one-way function in the $\mathcal{G}_{\mathsf{oRO}}$ model.

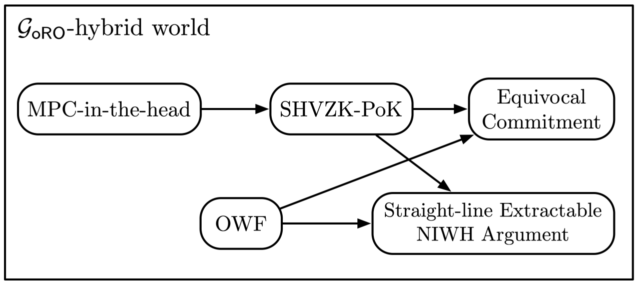
[1]Throughout this work, we do not consider the case of simultaneous rounds where two parties can send their messages to each other at the same round [GIS18, MR19].

To address the latter question, [CJS14] introduced a 5-round mechanism that enables the simulator to obtain the equivocation trapdoor in the $\mathcal{G}_{\mathsf{oRO}}$ model; whereas, [MRS17] proposed a more round-efficient (3-round) mechanism to do so. More precisely, [MRS17] let the receiver use a Non-Interactive Witness Indistinguishable (NIWI) argument to prove the knowledge of equivocation trapdoor w.r.t. the commitment key. The proof is sent together with the commitment key in the first round. Note that straight-line extractability is needed for this approach.

Inspired by the technique in [Pas03], our framework adopts a Non-Interactive Witness Hiding (NIWH) argument with straight-line extractability which can be constructed under MiniCrypt assumption in the $\mathcal{G}_{\mathsf{oRO}}$ model. Putting things together, we can obtain a GUC-secure commitment using only MiniCrypt assumptions. We present the technique roadmap of our framework in Figure 1.


$\mathcal{G}_{\mathsf{oRO}}$-model
Equivocal Commitment
GUC-secure Commitment
Straight-line Extractable NIWH Argument

(a) A general framework.


$\mathcal{G}_{\mathsf{oRO}}$-hybrid world
MPC-in-the-head
SHVZK-PoK
Equivocal Commitment
OWF
Straight-line Extractable NIWH Argument

(b) Instantiation of building blocks.

Figure 1: Technique Roadmap

Non-interactive equivocal commitment in MiniCrypt. As shown in [Dam02, MY04], it is possible build a non-interactive equivocal commitment from a 3-round public-coin Special Honest Verifier Zero-Knowledge (SHVZK) protocol with 2-special soundness. In the SHVZK protocols, the prover sends the message flow a in the first round, and the receiver sends a public-coin randomness e as the challenge in the second round. After receiving e, the prover computes and sends the response z in the last round. The technique of constructing non-interactive equivocal commitment can be summarized as follows. Let $\mathcal{R}$ be an NP relation. The receiver randomly samples a pair $(x, w) \in \mathcal{R}$ and sends x to the committer. To commit a message m, the committer invokes the SHVZK simulator for $x \in \mathcal{L}$, using m as the challenge. The simulator then outputs the simulated proof (a, z). The committer sends a to the receiver as its commitment message. To open the commitment, the committer can simply sends m, z to the receiver, who will accept it if and only if (a, m, z) is an accepting proof transcript. The equivocation trapdoor is w, which can be extracted from the straight-line extractor of NIWH as described in previous paragraph.

Since we aim to construct a commitment in MiniCrypt, in our construction, $\mathcal{R}$ is instantiated by a one-way function relation, i.e., $\mathsf{OWF}(x) = y$. Next, how to construct a 2-special sound SHVZK protocol in MiniCrypt? One possible approach is to use the "MPC-in-the-head" paradigm proposed by Ishai *et al.* [IKOS07]. Roughly speaking, the main idea is for the prover to simulate the execution of an n-party computation protocol that checks $(x, w) \in \mathcal{R}$, where x is the public input and w is the witness. The prover then commits to all views of the parties and sends the commitments to the verifier. After that, the verifier chooses a random subset of the parties and asks the prover to open their corresponding views. The verifier accepts the proof if the views are consistent. Unfortunately, to the best of our knowledge, none of the followups [GMO16, CDG+17, AHIV17, KKW18, dOT21] since the initial work of [IKOS07] can lead to a 2-special sound SHVZK protocol merely based on MiniCrypt assumptions. To address this issue, we propose a new technique that can construct a 2-special sound protocol in the $\mathcal{G}_{\mathsf{oRO}}$ model (cf. Section 4.2.1).

Towards a complete picture. In terms of the $\mathcal{G}_{\mathsf{oRO}}$, our work gives a complete answer to our questions: we show there exists *no* 2-round GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model (cf. Section 3), and present a 3-round (round-optimal) GUC-secure commitment using only MiniCrypt assumptions in the $\mathcal{G}_{\mathsf{oRO}}$ model (cf. Section 4). Moreover, it is known that GUC-secure commitment does not exist in the $\mathcal{G}_{\mathsf{sRO}}$ model [CDPW07], and round-optimal GUC-secure commitment can be constructed without further assumptions in the $\mathcal{G}_{\mathsf{poRO}}$

model [CDG$^+$18]. What about the $\mathcal{G}_{\mathsf{pRO}}$? In this work, we also show some impossibility result: there exists *no* GUC-secure commitment with one-round committing phase in the $\mathcal{G}_{\mathsf{pRO}}$ model (cf. Appendix B). However, the feasibility of round-optimal GUC-secure commitment via MiniCrypt assumptions in the $\mathcal{G}_{\mathsf{pRO}}$ model remains an open question.

Further investigation and future directions. We mainly focus on the commitment in this work. One may also wonder the lower bounds of the round complexity of other cryptographic primitives such as ZK, OT, etc. In fact, it is already known that there exists no NIZK in the observable RO model [Pas03]. What about the ZK proofs in the $\mathcal{G}_{\mathsf{pRO}}$ model? In this work, we show that our impossibility result can be extended to ZK proofs in the $\mathcal{G}_{\mathsf{pRO}}$ model: there exists no non-trivial GUC-secure NIZK protocols in the $\mathcal{G}_{\mathsf{pRO}}$ model (cf. Appendix C).

1.2 Related work

It terms of UC security with local setups, non-interactive commitments (1 round for the committing phase and 1 round for the opening phase) can be constructed under various setup assumptions. For instance, Canetti and Fischlin gave a candidate in the CRS model [CF01]; Hofheinz and Müller-Quade suggested a candidate in the RO model [HM04].

As for UC security with global setups, it is still unclear if it is possible to construct a non-interactive GUC-secure commitment, and very few work, e.g., [DSW08] is dedicated to this research area. In [CDPW07], Canetti *et al.* showed that it is impossible to construct a GUC-secure commitment merely relying on local CRS/RO functionalities; they further proposed a 7-round GUC-secure commitment protocol in the Argumented CRS (ACRS) model. Later, Dodis *et al.* proved that there exist *no* GUC-secure commitment with one-round commitment phase in the ACRS model against adaptive adversaries [DSW08]. Note that their impossibility result can be extended to any other global setup whose output depends on the program ID (pid) of the querying party, but not the session ID (sid), such as the Key Registration of Knowledge (KRK) model [BCNP04]. To bypass this impossibility result, $\mathcal{G}_{\mathsf{oRO}}$,$\mathcal{G}_{\mathsf{pRO}}$ and $\mathcal{G}_{\mathsf{poRO}}$ are proposed; the output of those setup functionalities depends on the session ID (sid).

Focusing on commitments in the $\mathcal{G}_{\mathsf{oRO}}$, Canetti *et al.* proposed a 5-round GUC-secure commitment [CJS14]. Later, Mohassel *et al.* gave a $(1+2)$-round GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model, where the committer and the receiver needs to have an additional one-round setup phase followed by a 2-round commitment [MRS17]. Note that their construction also employed Pedersen commitment and thus in the Crypto-Mania [Imp95] world. Byali *et al.* gave a 2-round GUC-secure commitment construction in the CRS and $\mathcal{G}_{\mathsf{oRO}}$ hybrid model [BPRS17]. Following Byali *et al.* paradigm, GUC-secure ZK protocols [GKPS18, LR22] can also be constructed in the CRS and $\mathcal{G}_{\mathsf{oRO}}$ hybrid model. With regard to post-quantum security, [BGM19] gave a 5-round lattice-based GUC-secure commitment and [Bra21] gave a 6-round code-based GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model.

In respect of the $\mathcal{G}_{\mathsf{pRO}}$ and the $\mathcal{G}_{\mathsf{poRO}}$, Camenisch *et al.* proposed a 3-round GUC-secure commitment from CDH assumption in the $\mathcal{G}_{\mathsf{pRO}}$ model and an information-theoretical non-interactive GUC-secure commitment in the $\mathcal{G}_{\mathsf{poRO}}$ model [CDG$^+$18]. In the following, Canetti *et al.* proposed a 2-round OT adaptive-secure OT from DDH assumption in the $\mathcal{G}_{\mathsf{pRO}}$ model [CSW20], but their protocol is only UC-secure. Baum *et al.* constructed a GUC-secure commitment scheme that is additively homomorphic in the $\mathcal{G}_{\mathsf{poRO}}$ model [BDD20].

2 Preliminaries

2.1 Notations

Let $\lambda \in \mathbb{N}$ be the security parameter. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large λ, it holds that $\mu(\lambda) < \frac{1}{p(\lambda)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$ for some $n \in \mathbb{N}$. For an NP relation $\mathcal{R}$, we denote by $\mathcal{L}$ its associate language, i.e. $\mathcal{L} = \{x \mid \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$. We say that two distribution ensembles $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ are identical (resp. computationally indistinguishable), denoted by $X \equiv Y$

(resp. $X \overset{c}{\approx} Y$), if for any unbounded (resp. PPT) distinguisher D there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[D(X_\lambda) = 1] - \Pr[D(Y_\lambda) = 1]| = 0$ (resp. $\mathsf{negl}(\lambda)$).

2.2 Universal Composability

UC. The UC framework proposed by Canetti [Can01] lays down a solid foundation for designing and analyzing protocols secure against attacks in an arbitrary network execution environment. Roughly speaking, in the UC framework, protocols are carried out over multiple interconnected parties. Every party is identified by the unique pair $(\mathsf{pid}, \mathsf{sid})$, where pid is the Program ID (PID) and sid is the Session ID (SID). Let $\mathcal{A}$ be the adversary who can control the network and corrupt the parties. When a party is corrupted, the adversary $\mathcal{A}$ receives its private input and its internal state. We say a protocol is *terminating* if it can terminate in polynomial time, and we only consider terminating protocols in this work.

We call a protocol, the one for which we want to prove security, challenge protocol. A challenge protocol Π is a UC-secure realization of a functionality $\mathcal{F}$, if it satisfies that for every PPT adversary $\mathcal{A}$ attacking an execution of Π, there is another PPT adversary $\mathcal{S}$—known as the simulator—attacking the ideal process that interacts with $\mathcal{F}$ (by corrupting the same set of parties), such that the executions of Π with $\mathcal{A}$ and that of $\mathcal{F}$ with $\mathcal{S}$ makes no difference to any PPT network execution environment $\mathcal{Z}$.

The ideal world execution. In the ideal world, the set of parties $\mathcal{P} := \{P_1, \ldots, P_n\}$ only communicate with an ideal functionality $\mathcal{F}$ and the simulator $\mathcal{S}$. The corrupted parties are controlled by the simulator $\mathcal{S}$. The output of the environment $\mathcal{Z}$ in this execution is denoted by $\mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

The real world execution. In the real world, the set of parties $\mathcal{P} := \{P_1, \ldots, P_n\}$ communicate with each other and the adversary $\mathcal{A}$ to run the protocol Π. The corrupted parties are controlled by the adversary $\mathcal{A}$. The output of the environment $\mathcal{Z}$ in this execution is denoted by $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$.

Definition 1. *We say a protocol Π UC-realizes functionality $\mathcal{F}$, if for any PPT environment $\mathcal{Z}$ and any PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ s.t.* $\mathsf{EXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

In order to conceptually modularize the design of the protocols, the notion of "hybrid world" is introduced. A protocol Π is said to be realized "in the $\mathcal{G}$ hybrid world" if Π invokes the ideal functionality $\mathcal{G}$ as a subroutine.

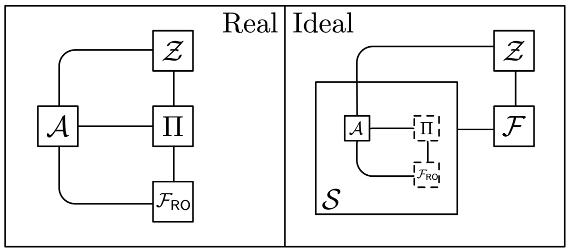
Definition 2. *We say protocol Π UC-realizes functionality $\mathcal{F}$ in the $\mathcal{G}$ hybrid world, if for any PPT environment $\mathcal{Z}$ and any PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ s.t.* $\mathsf{EXEC}^{\mathcal{G}}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

Furthermore, in the UC framework, the environment $\mathcal{Z}$ cannot have the direct access to $\mathcal{G}$, but it can do so through the adversary. Namely, in the real world, the adversary $\mathcal{A}$ can access the ideal functionality $\mathcal{G}$ directly, and $\mathcal{A}$ queries $\mathcal{G}$ for $\mathcal{Z}$ and forwards the answers; analogously, in the ideal world, $\mathcal{Z}$ can query $\mathcal{G}$ through the simulator $\mathcal{S}$. This implicitly means that $\mathcal{G}$ is local to the challenge protocol instance. This allows the simulator $\mathcal{S}$ to simulate $\mathcal{G}$ in the ideal world as long as it "looks" indistinguishable from $\mathcal{G}$ hybrid world.

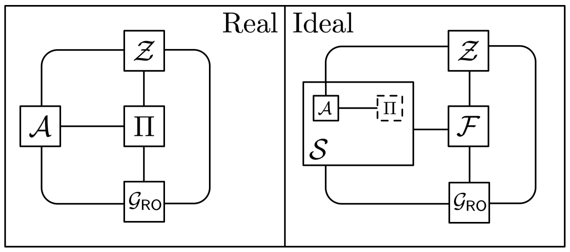
Generalized UC. In the basic UC, the environment $\mathcal{Z}$ is constrained: it cannot have the direct access to the setup. It means that the setup is not global. This assumption might be impractical in real life applications where it is more plausible that there is a global setup published and used by many protocols.

Motivated by solving problems caused by modeling setup as a local subroutine, Canetti *et al.* introduced Generalized UC (GUC) which can be used for properly analyzing concurrent execution of protocols in the presence of global setup [CDPW07]. In the GUC framework, the environment $\mathcal{Z}$ is unconstrained: $\mathcal{Z}$ is allowed to access the setup directly without going through the simulator/adversary and invoke arbitrary protocols alongside the challenge protocol. Furthermore, the setup can be modeled as a *shared functionality* that can communicate with more than one protocol sessions. Let the output of the unconstrained PPT environment $\mathcal{Z}$ in the real world (resp. ideal world) execution be denoted by $\mathsf{GEXEC}_{\Pi,\mathcal{A},\mathcal{Z}}$ (resp. $\mathsf{GEXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$).

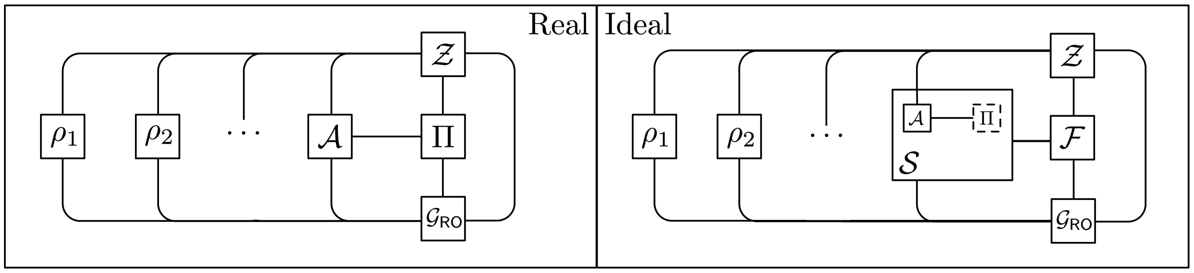
Definition 3. *We say a protocol Π GUC-realizes functionality $\mathcal{F}$, if for any unconstrained PPT environment $\mathcal{Z}$ and any PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ s.t.* $\mathsf{GEXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{GEXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.



(a) Basic UC: the simulator $\mathcal{S}$ simulates the local $\mathcal{F}_{\mathsf{RO}}$ and has full control.



(b) EUC: the global $\mathcal{G}_{\mathsf{RO}}$ is external to the simulator, and the environment $\mathcal{Z}$ is $\mathcal{G}$-externalized constrained.



(c) GUC: the global $\mathcal{G}_{\mathsf{RO}}$ is external to the simulator, and the environment $\mathcal{Z}$ not only has direct access to $\mathcal{G}_{\mathsf{RO}}$, but also invokes arbitraty protocols $\rho_1, \rho_2, \cdots$ alongside the challenge protocol Π.

Figure 2: Comparison of Basic UC, GUC and EUC.

Since the unconstrained environment $\mathcal{Z}$ is given a high-level of flexibility: $\mathcal{Z}$ is allowed to invoke arbitrary protocols in parallel with the challenge protocol. This makes it extremely hard to prove the GUC security. Therefore, a simplified framework called Externalized UC (EUC) is introduced in [CDPW07]. In the EUC framework, the environment $\mathcal{Z}$ has direct access to the shared functionality $\mathcal{G}$ but does not initiate any new protocol sessions except the challenge protocol session. We call such an environment is $\mathcal{G}$-externalized constrained. We say a protocol Π is $\mathcal{G}$-subroutine respecting if it only shares state information via a single shared functionality $\mathcal{G}$. We take RO models as an example, and present the comparison of basic UC, GUC and EUC in Figure 2.

Definition 4. *Let the protocol Π be $\mathcal{G}$-subroutine respecting. We say a protocol Π EUC-realizes functionality $\mathcal{F}$ with respect to shared functionality $\mathcal{G}$, if for any PPT $\mathcal{G}$-externalized constrained environment $\mathcal{Z}$ and any PPT adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ s.t.* $\mathsf{EXEC}^{\mathcal{G}}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}^{\mathcal{G}}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

In [CDPW07], Canetti *et al.* showed that for any $\mathcal{G}$-subroutine respecting protocol Π, proving Π EUC-realizes $\mathcal{F}$ with respect to $\mathcal{G}$ is equivalent to proving Π GUC-realizes $\mathcal{F}$. Therefore, when we want to prove the GUC security of a protocol, we always turn to EUC security for the sake of simplicity.

2.3 The Global Random Oracle Models

In this section, we review four well-known Global Random Oracle (GRO) models: (i) Global Strict Random Oracle (GSRO) model proposed by Canetti *et al.* in [CJS14], which does not give any extra power to anyone; (ii) Global Observable Random Oracle (GORO) model proposed by Canetti *et al.* in [CJS14], which grants the ideal world simulator access to the list of illegitimate queries (to be defined later); (iii) Global Programmable Random Oracle (GPRO) model proposed by Camenisch *et al.* in [CDG+18], which allows the simulator/adversary to program on unqueried points; (iv) Global Programmable and Observable Random Oracle (GPORO) model proposed by Camenisch *et al.* in [CDG+18], which provides both programmability and observability. We present the formal description of all the global random oracle models mentioned above in Figure 3, and the relation of these models in Figure 4.

The GSRO model. The GSRO model $\mathcal{G}_{\mathsf{sRO}}$ is a natural extension of local RO model $\mathcal{F}_{\mathsf{RO}}$: as depicted in Figure 3(a), upon receiving (Query, sid, x) from any party, $\mathcal{G}_{\mathsf{sRO}}$ first checks if the query (sid, x) has been

Shared Functionality $\mathcal{G}_{\mathsf{sRO}}$

The functionality interacts with a set of party $\mathcal{P} := \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$. It is parameterized by the input/output length $\ell_{\mathsf{in}}(\lambda)$ and $\ell_{\mathsf{out}}(\lambda)$. It maintains an initially empty list List.

- **Query.** Upon receiving $(\textsc{Query}, s, x)$ from a party $P_i := (\mathsf{pid}, \mathsf{sid}) \in \mathcal{P}$ or the adversary $\mathcal{S}$:
  - Find v such that $(s, x, v) \in \mathsf{List}$. If there is no such v exists, select an uniformly random $v \in \{0,1\}^{\ell_{\mathsf{out}}(\lambda)}$ and record the tuple (s, x, v) in List.
  - Return $(\textsc{QueryConfirm}, s, v)$ to the requestor.

(a) The Global Strict Random Oracle Model $\mathcal{G}_{\mathsf{sRO}}$

Shared Functionality $\mathcal{G}_{\mathsf{oRO}}$

The functionality interacts with a set of party $\mathcal{P} := \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$. It is parameterized by the input/output length $\ell_{\mathsf{in}}(\lambda)$ and $\ell_{\mathsf{out}}(\lambda)$, and a list of ideal functionality programs $\bar{\mathcal{F}}$. It maintains an initially empty list List.

- **Query.** Same as $\mathcal{G}_{\mathsf{sRO}}$ depicted in Figure 3(a), except when $\mathsf{sid} \neq s$, add the tuple (s, x, v) to the (initially empty) list of illegitimate queries for SID s, which we denote by $\mathcal{Q}_s$.
- **Observe.** Upon receiving a request from an instance of an ideal functionality in the list $\bar{\mathcal{F}}$, with SID s, return to this instance the list of illegitimate queries $\mathcal{Q}_s$ for SID s.

(b) The Global Observable Random Oracle Model $\mathcal{G}_{\mathsf{oRO}}$

Shared Functionality $\mathcal{G}_{\mathsf{pRO}}$

The functionality interacts with a set of party $\mathcal{P} := \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$. It is parameterized by the input/output length $\ell_{\mathsf{in}}(\lambda)$ and $\ell_{\mathsf{out}}(\lambda)$. It maintains initially empty lists $\mathsf{List}, \mathsf{Prog}$.

- **Query.** Same as $\mathcal{G}_{\mathsf{sRO}}$ depicted in Figure 3(a).
- **Program.** Upon receiving $(\textsc{Program}, \mathsf{sid}, x, v)$ with $v \in \{0,1\}^{\ell_{\mathsf{out}}(\lambda)}$ from an adversary $\mathcal{S}$:
  - If $\exists v' \in \{0,1\}^{\ell_{\mathsf{out}}(\lambda)}$ such that $(\mathsf{sid}, x, v') \in \mathsf{List}$ and $v \neq v'$, ignore this input.
  - Set $\mathsf{List} := \mathsf{List} \cup \{(\mathsf{sid}, x, v)\}$ and $\mathsf{Prog} := \mathsf{Prog} \cup \{(\mathsf{sid}, x)\}$.
  - Return $(\textsc{ProgramConfirm}, \mathsf{sid})$ to $\mathcal{S}$.
- **IsProgramed.** Upon receiving $(\textsc{IsProgramed}, \mathsf{sid}', x)$ from a party $P_i := (\mathsf{pid}, \mathsf{sid})$ or the adversary $\mathcal{S}$:
  - If the input was given by $P_i := (\mathsf{pid}, \mathsf{sid})$ and $\mathsf{sid} \neq \mathsf{sid}'$, ignore this input.
  - Set $b \leftarrow (\mathsf{sid}', x) \in \mathsf{Prog}$, and return $(\textsc{IsProgramed}, \mathsf{sid}', b)$ to the requester.

(c) The Global Programmable Random Oracle Model $\mathcal{G}_{\mathsf{pRO}}$

Shared Functionality $\mathcal{G}_{\mathsf{poRO}}$

The functionality interacts with a set of party $\mathcal{P} := \{P_1, \ldots, P_n\}$ and an adversary $\mathcal{S}$. It is parameterized by the input/output length $\ell_{\mathsf{in}}(\lambda)$ and $\ell_{\mathsf{out}}(\lambda)$, and a list of ideal functionality programs $\bar{\mathcal{F}}$. It maintains initially empty lists $\mathsf{List}, \mathsf{Prog}$.

- **Query/Observe.** Same as $\mathcal{G}_{\mathsf{oRO}}$ depicted in Figure 3(b).
- **Program/IsProgramed.** Same as $\mathcal{G}_{\mathsf{pRO}}$ depicted in Figure 3(c).

(d) The Global Programmable and Observable Random Oracle Model $\mathcal{G}_{\mathsf{poRO}}$

Figure 3: The Global Random Oracle Models.

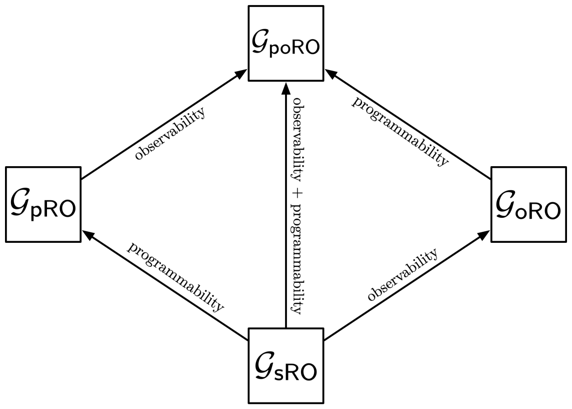

$\mathcal{G}_{\mathsf{poRO}}$
$\mathcal{G}_{\mathsf{pRO}}$
$\mathcal{G}_{\mathsf{oRO}}$
$\mathcal{G}_{\mathsf{sRO}}$
observability
programmability
observability + programmability
programmability
observability

Figure 4: Relation of the Global Random Oracle Models

queried before. If not, $\mathcal{G}_{\mathsf{sRO}}$ answers with a random value of pre-specified length, that is $v \in \{0,1\}^{\ell_{\mathsf{out}}(\lambda)}$, and records the tuple (sid, x, v); otherwise, the previously chosen value v is returned again even if the earlier query was made by another party. The sad truth is that Canetti *et al.* remarked that $\mathcal{G}_{\mathsf{sRO}}$ does not suffice to GUC-realizes commitment functionality. Therefore, stronger variant global random oracle models are needed to realize non-trivial functionalities.

The GORO model. Compared to $\mathcal{G}_{\mathsf{sRO}}$, the GORO model $\mathcal{G}_{\mathsf{oRO}}$ provides additionally observability. More precisely, some of the queries can be marked as "illegitimate" and potentially disclosed to the simulator. As depicted in Figure 3(b), for any query (s, x) from any party $P := (\mathsf{pid}, \mathsf{sid})$ where s is the content of the SID field, if $s \neq \mathsf{sid}$, then this query is considered "illegitimate". After that, $\mathcal{G}_{\mathsf{oRO}}$ adds the tuple (s, x, v) to the list of illegitimate queries for SID s, which we denote as $\mathcal{Q}_s$. The illegitimate queries $\mathcal{Q}_s$ may be disclosed to the instance of ideal functionality whose SID is the one of the illegitimate queries. Then the ideal functionality instance leaks the illegitimate queries to the simulator.

The GPRO model. Compared to $\mathcal{G}_{\mathsf{sRO}}$, the GPRO model $\mathcal{G}_{\mathsf{pRO}}$ additionally allows simulator/adversary to program the global random oracle on unqueried points. As depicted in Figure 3(c), upon receiving $(\textsc{Program}, \mathsf{sid}, x, v)$ from the simulator/adversary, $\mathcal{G}_{\mathsf{pRO}}$ first checks if (sid, x) has been queried before. If not, $\mathcal{G}_{\mathsf{pRO}}$ stores (sid, x, v) in the query-answer lists. Any honest party can check whether a point has been programmed or not by sending the $(\textsc{IsProgramed}, \mathsf{sid}, x)$ command to $\mathcal{G}_{\mathsf{pRO}}$. Thus, in the real world, the programmed points can always be detected. However, in the ideal world, the simulator $\mathcal{S}$ can successfully program the random oracle without being detected since it can always return $(\textsc{IsProgramed}, \mathsf{sid}, 0)$ when the adversary invokes $(\textsc{IsProgramed}, \mathsf{sid}, x)$ to verify whether a point x has been programmed or not.

The GPORO model. If we combine the GORO model and GPRO model together, we obtain the GPORO model $\mathcal{G}_{\mathsf{poRO}}$ which is depicted in Figure 3(d). To the best of our knowledge, the GPORO model is the most powerful GRO model that enables efficient composable protocols in the GUC framework. For example, Camenisch *et al.* gave an efficient non-interactive GUC-secure commitment protocol in the GPORO model [CDG+18].

Remark 1. *Camenisch et al. remarked that when one uses the (distinguishing) environment in a cryptographic reduction, one can have full control over the shared functionality [CDG+18]. More precisely, as depicted in Figure 5, the reduction algorithm $\mathcal{B}$ simulates the complete view of the environment $\mathcal{Z}$ including the shared functionality $\mathcal{G}$, thus $\mathcal{B}$ has full control of $\mathcal{G}$.*

2.4 One-Way Functions

One-Way Function (OWF) is the minimal cryptographic primitive. Informally, a one-way function $\mathsf{OWF} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is easy to compute but hard to invert which defined as follows:

Definition 5 (Easy to Compute)**.** *We say it is easy to compute if for any $x \in \{0,1\}^\lambda$, there exists a PPT algorithm that on input x outputs $\mathsf{OWF}(x)$.*

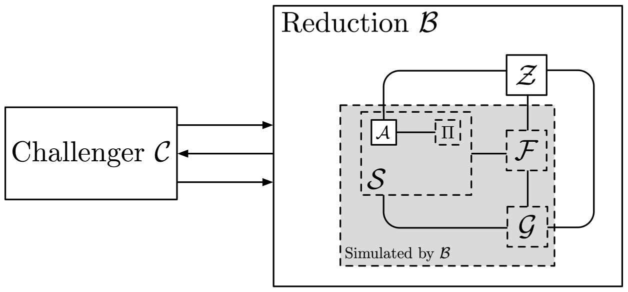


Figure 5: In order to play against the external challenger $\mathcal{C}$, reduction algorithm $\mathcal{B}$ simulates everything (marked as gray) including shared functionality $\mathcal{G}$, starts the protocol Π with the real world adversary $\mathcal{A}$/environment $\mathcal{Z}$ by running $\mathcal{A}/\mathcal{Z}$ internally as black-box.

Definition 6 (Hard to Invert)**.** *We say it is Hard to Invert (HI) if for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *such that* $\Pr[\textsc{expt}^{\mathsf{HI}}(\mathcal{A},\lambda)=1] \leq \mathsf{negl}(\lambda)$*, where* $\textsc{expt}^{\mathsf{HI}}(\mathcal{A},\lambda)$ *is defined as follows:*

- *$\mathcal{A}$ interacts with a game challenger, denoted as $\mathcal{C}$.*
- *$\mathcal{C}$ selects a random $x \in \{0,1\}^\lambda$, computes $y \leftarrow \mathsf{OWF}(x)$, and hands y to $\mathcal{A}$.*
- *$\mathcal{A}$ computes x' and sends x' to $\mathcal{C}$.*
- *$\mathcal{C}$ checks $\mathsf{OWF}(x') = y$. If the check passes, $\mathcal{A}$ wins; otherwise, $\mathcal{A}$ loses.*

Here we define by $\mathbf{Adv}^{\mathsf{HI}}(\mathcal{A},\lambda) := \Pr[\textsc{expt}^{\mathsf{HI}}(\mathcal{A},\lambda)=1]$ *the advantage of an adversary $\mathcal{A}$.*

2.5 Interactive Proof/Argument System

We denote by $\langle P(w), V\rangle(x)$ the output bit b of the verifier V when interacting with the prover P who has a private input w on common input x, and $b = 1$ indicates acceptance while $b = 0$ indicates rejection. Formally, a pair of PPT randomized algorithms $\langle P, V\rangle$ is called an interactive argument system for the language $\mathcal{L}$ if it satisfies the following properties:

Definition 7 (Completeness)**.** *For any $(x, w) \in \mathcal{R}$, we say it is complete if there exists a negligible function* negl *such that*

$$\Pr[\langle P(w), V\rangle(x) = 1] = 1 - \mathsf{negl}(\lambda)$$

Definition 8 (Computational Soundness)**.** *For any $x \notin \mathcal{L}$, we say it is computational sound if for any PPT P^*, there exists a negligible function* negl *such that*

$$\Pr[\langle P^*, V\rangle(x) = 1] = \mathsf{negl}(\lambda)$$

We say an interactive argument $\langle P, V\rangle$ is an interactive proof for the language $\mathcal{L}$ if the following stronger soundness property holds:

Definition 9 (Soundness)**.** *For any $x \notin \mathcal{L}$, we say it is sound if for any P^*, there exists a negligible function* negl *such that*

$$\Pr[\langle P^*, V\rangle(x) = 1] = \mathsf{negl}(\lambda)$$

2.6 SHVZK Protocols

We define a three round public coin Special Honest Verifier Zero-Knowledge (SHVZK) protocol $\Pi_{\mathsf{SHVZK}} := (\mathcal{P}_1, \mathcal{P}_2, \mathcal{V}_1, \mathcal{V}_2)$ with the following particular structure:

- Protocol:

1. $\mathcal{P}_1(x, w, r)$ takes input as a statement-witness pair (x, w), a random coin r, and sends the first flow a.
2. $\mathcal{V}_1(1^\lambda)$ takes input as a security parameter λ, and sends a random challenge e.
3. $\mathcal{P}_2(x, w, r, e)$ takes input as a statement-witness pair (x, w), a random coin r, a challenge e, and sends a response $z := f(r, a, e)$, where f is a public function.

- Verification: $\mathcal{V}_2(x, a, e, z)$ takes input as a statement x, the first flow a, a challenge e, a response z, and outputs a bit b indicating acceptance or not.

A Π_{SHVZK} protocol is an interactive proof system that satisfies the following property.

Definition 10 (Special Honest Verifier Zero-Knowledge)**.** *For any $(x, w) \in \mathcal{R}$, we say it has Special Honest Verifier Zero-Knowledge (SHVZK) if there exists a PPT simulator* Sim *such that for any PPT adversary $\mathcal{A}$, there exists a negligible function* negl *s.t.* $\Pr[\text{EXPT}^{\mathsf{SHVZK}}(\mathcal{A}, \lambda) = 1] - \frac{1}{2} \leq \mathsf{negl}(\lambda)$*, where* $\text{EXPT}^{\mathsf{SHVZK}}(\mathcal{A}, \lambda)$ *is defined as follows:*

- *$\mathcal{A}$ interacts with a game challenger, denoted as $\mathcal{C}$.*
- *$\mathcal{A}$ computes $(x, w) \in \mathcal{R}$ along with a challenge e, and sends (x, w, e) to $\mathcal{C}$.*
- *$\mathcal{C}$ selects a random coin r and a random bit $b \in \{0, 1\}$, and computes:*
  - *If $b = 0$: $a \leftarrow \Pi_{\mathsf{SHVZK}}.\mathcal{P}_1(x, w, r)$, $z \leftarrow \Pi_{\mathsf{SHVZK}}.\mathcal{P}_2(x, w, r, e)$.*
  - *If $b = 1$: $(a, z) \leftarrow \Pi_{\mathsf{SHVZK}}.\mathsf{Sim}(x, e, r)$.*

  $\mathcal{C}$ then sends (a, z) to $\mathcal{A}$.
- *$\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. If $b = b'$, output 1; otherwise, output 0.*

Denote by $\mathbf{Adv}^{\mathsf{SHVZK}}(\mathcal{A}, \lambda) = \Pr[\text{EXPT}^{\mathsf{SHVZK}}(\mathcal{A}, \lambda) = 1] - \frac{1}{2}$ *the advantage of $\mathcal{A}$.*

In our work, we let our Π_{SHVZK} protocol additionally satisfy the following property.

Definition 11 (k-Special Soundness)**.** *For any x, we say it has k-special soundness (k-SS) if $k \in \mathbb{N}$ and $k \geq 2$, and there exists a PPT extractor* Ext *such that for any PPT adversary $\mathcal{A}$,* $\Pr[\text{EXPT}^{\mathsf{SS}}(\mathcal{A}, \lambda) = 1] = 0$ *always holds, where* $\text{EXPT}^{\mathsf{SS}}(\mathcal{A}, \lambda)$ *is defined as follows:*

- *$\mathcal{A}$ interacts with a game challenger, denoted as $\mathcal{C}$.*
- *$\mathcal{A}$ computes x and a set of k accepting transcripts $\{(a, e_i, z_i)\}_{i\in[k]}$ with $e_i \neq e_j$ where $i \neq j$, and sends $(x, \{(a, e_i, z_i)\}_{i\in[k]})$ to $\mathcal{C}$.*
- *$\mathcal{C}$ checks if: (i) $e_i \neq e_j$ where $i \neq j$; (ii) $\forall i \in [k] : \Pi_{\mathsf{SHVZK}}.\mathcal{V}_2(x, a, e_i, z_i) = 1$. If so, $\mathcal{C}$ runs $w \leftarrow \Pi_{\mathsf{SHVZK}}.\mathsf{Ext}(x, \{(a, e_i, z_i)\}_{i\in[k]})$. If $\Pi_{\mathsf{SHVZK}}.\mathsf{Ext}$ outputs a valid witness w such that $\mathcal{R}(x, w) = 1$, output 0; otherwise, output 1.*

Denote by $\mathbf{Adv}^{\mathsf{SS}}(\mathcal{A}, \lambda) := \Pr[\text{EXPT}^{\mathsf{SS}}(\mathcal{A}, \lambda) = 1]$ *the advantage of $\mathcal{A}$.*

2.7 Non-Interactive Witness Hiding

Witness Hiding (WH) interactive proofs were introduced by Feige and Shamir in [FS90], and we use the non-interactive setting here [KZ20]. That is, the prover P on input (x, w) to generate the proof π and send π to the verifier V. Given the proof π, the verifier V cannot compute any new witness that V does not know before the interaction. Formally, we present the following definition:

Definition 12 (Hard Instance Ensembles)**.** *Let $\mathcal{R}$ be an NP relation, and $\mathcal{L}$ be its associate language, and $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ be a probability ensemble s.t. $\mathcal{X}_\lambda$ ranges over $\mathcal{L} \cap \{0, 1\}^\lambda$. We say that $\mathcal{X}$ is hard for $\mathcal{R}$ if for any PPT $\mathcal{A}$ and any $x \in \mathcal{X}$, there exists a negligible function* negl *s.t.* $\Pr[(x, \mathcal{A}(x)) \in \mathcal{R}] = \mathsf{negl}(\lambda)$.

Definition 13 (Witness Hiding). *Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ be a hard instance ensemble for $\mathcal{R}$. We say a proof system $\langle P, V\rangle$ is witness hiding for $\mathcal{R}$ under the instance ensemble $\mathcal{X}$ if for any PPT $\mathcal{A}$ and any $(x, w) \in \mathcal{R}$ with $x \in \mathcal{X}$, there exists a negligible function* negl *s.t.* $\Pr[\textsc{expt}^{\mathsf{WH}}(\mathcal{A}, \lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\textsc{expt}^{\mathsf{WH}}(\mathcal{A}, \lambda)$ *is defined as follows:*

- *$\mathcal{A}$ interacts with a game challenger, denoted as $\mathcal{C}$.*
- *$\mathcal{C}$ computes $\pi \leftarrow \Pi_{\mathsf{NIWH}}.P(x, w)$, and sends (x, π) to $\mathcal{A}$.*
- *$\mathcal{A}$ extracts a witness w^* from (x, π), and sends w^* to $\mathcal{C}$.*
- *$\mathcal{C}$ checks if $(x, w^*) \in \mathcal{R}$ holds. If so, output 1; otherwise, output 0*

Denote by $\mathbf{Adv}^{\mathsf{WH}}(\mathcal{A}, \lambda) = \Pr[\textsc{expt}^{\mathsf{WH}}(\mathcal{A}, \lambda) = 1]$ *the advantage of $\mathcal{A}$. We say a proof system $\langle P, V\rangle$ is witness hiding for $\mathcal{R}$ if it is witness hiding under all hard-instance ensembles $\mathcal{X}$ for $\mathcal{R}$.*

2.8 Equivocal Commitment

An equivocal commitment $\mathsf{ECom} := (\mathsf{KeyGen}, \mathsf{KeyVer}, \mathsf{Commit}, \mathsf{ComVer}, \mathsf{EquCom}, \mathsf{Equiv})$ allows the committer to generate the commitment message c to any value m using the commitment key ck and the randomness r. Later, the committer can open c to m by sending the the opening d to the receiver who verifies it. Furthermore, if the committer obtains the equivocation trapdoor td with respect to the ck, he can generate the equivocal commitment $\tilde{c}$, later open $\tilde{c}$ to any message $\tilde{m}$. Formally, the equivocal commitment has the following algorithms:

- $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ takes input as the security parameter λ, and outputs a commitment key ck and an equivocation trapdoor td.
- $b \leftarrow \mathsf{KeyVer}(\mathsf{ck}, \mathsf{td})$ takes input as a commitment key ck and an equivocation trapdoor td. It outputs a bit b indicating acceptance or rejection.
- $(c, d) \leftarrow \mathsf{Commit}(\mathsf{ck}, m; r)$ takes input as a commitment key ck, a message m and a randomness r. It outputs the commitment c and the opening d. We assume that there exists a deterministic algorithm that can extract m from d. When r is not important, we use $\mathsf{Commit}(\mathsf{ck}, m)$ for simplicity.
- $b \leftarrow \mathsf{ComVer}(\mathsf{ck}, c, d)$ takes input as a commitment key ck, and a commitment-opening pair (c, d). It outputs a bit b indicating acceptance or rejection.
- $(\tilde{c}, \mathsf{st}) \leftarrow \mathsf{EquCom}(\mathsf{ck}, \mathsf{td}; r)$ takes input as a commitment key ck, an equivocation trapdoor td, and a randomness r. It outputs a commitment $\tilde{c}$ and a state st. When r is not important, we use $\mathsf{EquCom}(\mathsf{ck}, \mathsf{td})$ for simplicity.
- $\tilde{d} \leftarrow \mathsf{Equiv}(\mathsf{ck}, \mathsf{td}, \tilde{c}, \mathsf{st}, \tilde{m})$ takes input as a commitment key ck, an equivocation trapdoor td, a commitment $\tilde{c}$, a state st, and an arbitrary message $\tilde{m}$ for which equivocation is required. It outputs an opening $\tilde{d}$.

The equivocal commitment should satisfy the following definition:

Definition 14 (Correctness). *For any message m, we say it is correct if*

$$\Pr[(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda), (c, d) \leftarrow \mathsf{Commit}(\mathsf{ck}, m) : \mathsf{ComVer}(\mathsf{ck}, c, d) = 1] = 1$$

Definition 15 (Perfect Hiding). *We say it is perfect hiding if for any adversary $\mathcal{A}$ s.t.* $\Pr[\textsc{expt}^{\mathsf{HIDE}}(\mathcal{A}, \lambda) = 1] = \frac{1}{2}$*, where* $\textsc{expt}^{\mathsf{HIDE}}(\mathcal{A}, \lambda)$ *is defined as follows:*

- *$\mathcal{A}$ interacts with a game challenger, denoted as $\mathcal{C}$.*
- *$\mathcal{C}$ generates $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends ck to $\mathcal{A}$.*
- *$\mathcal{A}$ selects two distinct message m_0, m_1, and sends them to $\mathcal{C}$.*

- *$\mathcal{C}$ picks a random bit $b \in \{0,1\}$, computes $c \leftarrow \mathsf{Commit}(\mathsf{ck}, m_b)$ to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs a bit b'. If $b' = b$, output 1; otherwise, output 0.*

Definition 16 (Binding)**.** *We say it is binding if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl s.t. $\Pr[\text{EXPT}^{\mathsf{BIND}}(\mathcal{A}, \lambda) = 1] \leq \mathsf{negl}(\lambda)$, where $\text{EXPT}^{\mathsf{BIND}}(\mathcal{A}, \lambda)$ is defined as follows:*

- *$\mathcal{A}$ interacts with a game challenger, denoted as $\mathcal{C}$.*
- *$\mathcal{C}$ generates $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends ck to $\mathcal{A}$.*
- *$\mathcal{A}$ computes two distinct openings d_0, d_1 for the same commitment c, and sends them to $\mathcal{C}$.*
- *$\mathcal{C}$ checks if $d_0 \neq d_1$ and $\mathsf{ComVer}(\mathsf{ck}, c, d_0) = \mathsf{ComVer}(\mathsf{ck}, c, d_1) = 1$ holds. If so, output 1; otherwise, output 0.*

We denote by $\mathbf{Adv}^{\mathsf{BIND}}(\mathcal{A}, \lambda) = \Pr[\text{EXPT}^{\mathsf{BIND}}(\mathcal{A}, \lambda) = 1]$ the advantage of $\mathcal{A}$.

Definition 17 (Equivocal)**.** *We say it is equivocal if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl s.t. $\Pr[\text{EXPT}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda) = 1] - \frac{1}{2} \leq \mathsf{negl}(\lambda)$, where $\text{EXPT}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda)$ is defined as follows:*

- *$\mathcal{A}$ interacts with a game challenger, denoted as $\mathcal{C}$.*
- *$\mathcal{A}$ generates $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends $(\mathsf{ck}, \mathsf{td})$ along with a message m to $\mathcal{C}$.*
- *$\mathcal{C}$ checks if $\mathsf{KeyVer}(\mathsf{ck}, \mathsf{td}) = 1$ holds. If so, $\mathcal{C}$ selects a random bit $b \in \{0,1\}$. $\mathcal{C}$ works as follows:*
  - *If $b = 0$: invoke $(c, d) \leftarrow \mathsf{Commit}(\mathsf{ck}, m)$, and send (c, d) to $\mathcal{A}$.*
  - *If $b = 1$: invoke $(\tilde{c}, \mathsf{st}) \leftarrow \mathsf{EquCom}(\mathsf{ck}, \mathsf{td})$ and $\tilde{d} \leftarrow \mathsf{Equiv}(\mathsf{ck}, \mathsf{td}, \tilde{c}, \mathsf{st}, m)$, and send $(\tilde{c}, \tilde{d})$ to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs a bit b'. If $b' = b$, output 1; otherwise, output 0.*

Denote by $\mathbf{Adv}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda) = \Pr[\text{EXPT}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda) = 1] - \frac{1}{2}$ the advantage of $\mathcal{A}$.

2.9 Secure Multi-Party Computation

Consider a public function $f : (\{0,1\}^\lambda)^{n+1} \to \{0,1\}^\lambda$, and let $P_1, \ldots, P_n$ be n parties modeled as PPT interactive machines. Each party P_i holds a private input $w_i \in \{0,1\}^\lambda$ and a public input $x \in \{0,1\}^\lambda$, and wants to compute $y = f(x, w)$, where $w := (w_1, \ldots, w_n)$. They communicate with each other using point-to-point secure channels, such as encrypted channels or OT-channels [Rab05], in the synchronous model. To achieve their goal, the parties jointly run a secure Multi-Party Computation (MPC) protocol Π_{MPC}. The protocol Π_{MPC} is specified via the next-message functions: there are multiple communication rounds, and in each round the party P_i sends into the channel a message that is computed as a deterministic function of the internal state of P_i (including private input w_i and random tape k_i) and the messages that P_i has received in the previous rounds. We define the view of the party P_i as the concatenation of the inputs x, w_i, the random tape k_i and all the messages received by P_i during the execution of Π_{MPC}. We denote by $\mathsf{view}_i(x, w_i)$ the view of P_i. Each channel defines a relation of consistency between views. For instance, in the plain model, we say $\mathsf{view}_i(x, w_i)$ and $\mathsf{view}_j(x, w_j)$ are consistent if the outgoing messages implicit in $\mathsf{view}_i(x, w_i)$ are identical to the incoming messages reported in $\mathsf{view}_j(x, w_j)$ and vice versa. Finally, all the views should yield the same output y, i.e. there are n functions $\Pi_{f,1}, \ldots, \Pi_{f,n}$ such that $y = \Pi_{f,i}(\mathsf{view}_i(x, w_i))$ for all $i \in [n]$. We note that, for our purpose of use, we require that every party P_i in the honest execution of Π_{MPC} has the same output y; in the general case, the output of P_i can be different from each other.

We consider security of MPC protocols in both the semi-honest and the malicious models. Here we turn to "property-paradigm" instead of "simulation-paradigm", since our purpose of introducing MPC is to apply "MPC-in-the-head" technique [IKOS07] to construct a stand-alone SHVZK-PoK protocol.

In the semi-honest model, the corrupted parties follow the instructions of the protocol, but are curious about the private information of other parties. Thus, the protocol needs to be designed in such a way that a corrupted P_i cannot infer information about w_j from its view $\mathsf{view}_i(x, w_i)$, where $j \neq i$. More formally, we have the following definitions.

Definition 18 (Correctness)**.** *For any inputs* $x, w := (w_1, \ldots, w_n)$ *and any random tape, we say* Π_{MPC} *realizes* f *with perfect correctness if* $\forall i \in [n] : \Pr[y = \Pi_{f,i}(\mathsf{view}_i(x, w_i))] = 1$*.*

We denote by $\mathsf{view}_T(x, w_1, \ldots, w_n)$ the joint view of players in set $T \subset [n]$ for the execution of Π_{MPC} on input $(x, w_1, \ldots, w_n)$. Consider a PPT simulator algorithm Sim that given the set $T \subset [n]$, the output of Π_{MPC} which realizes f on input $(x, w_1, \ldots, w_n)$ (i.e. $f(x, w_1, \ldots, w_n)$), and the input of parties in T (i.e. $x, (w_i)_{i \in T}$), it can output the simulated joint view of players in set T for the execution of Π_{MPC} on input $(x, w_1, \ldots, w_n)$ which we denote by $\mathsf{Sim}(T, x, (w_i)_{i \in T}, f(x, w_1, \ldots, w_n))$. With these notions, we have the following definition.

Definition 19 (t-Privacy)**.** *Let* $1 \leq t < n$*. We say* Π_{MPC} *realizes* f *with perfect* t*-privacy if it is perfect correct and for every set of corrupted parties* $T \subset [n]$ *satisfying* $|T| \leq t$*, there exists a PPT simulator* Sim *such that*

$$\mathsf{view}_T(x, w_1, \ldots, w_n) \equiv \mathsf{Sim}(T, x, (w_i)_{i \in T}, f(x, w_1, \ldots, w_n))$$

In the malicious model, the corrupted parties may behave arbitrarily. Therefore, we require the protocol Π_{MPC} be t-private as defined above, and moreover it should satisfy the following notion in the malicious model:

Definition 20 (t-Robustness)**.** *Let* $1 \leq t < n$*. We say* Π_{MPC} *realizes* f *with perfect* t*-robustness if it is perfect correct, and for any computationally unbounded malicious adversary corrupting a set* T *of at most* t *players, and for any inputs* $(x, w_1, \ldots, w_n)$*, the following holds: if there is no* $(w'_1, \ldots, w'_n)$ *such that* $f(x, w'_1, \ldots, w'_n) = 1$*, then the probability that some uncorrupted parties output 1 in an execution of* Π_{MPC} *in which the inputs of the honest parties are consistent with* $(x, w_1, \ldots, w_n)$ *is 0.*

3 Impossibility in the GORO Model

In this section, we show that it is impossible to construct 2-round GUC-secure commitment (one round for the committing phase and one round the for the opening phase) in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world against static adversaries. We first provide the formal description of transferable commitment functionality $\mathcal{F}_{\mathsf{tCOM}}$ from [CJS14] in Figure 6. The main difference with the normal commitment functionality is that in $\mathcal{F}_{\mathsf{tCOM}}$, the simulator can request the list of the illegitimate queries.

Functionality $\mathcal{F}_{\mathsf{tCOM}}$

The functionality interacts with two parties C, R and an adversary $\mathcal{S}$.

- Upon receiving $(\textsc{Commit}, \mathsf{sid}, C, R, m)$ from C, do:
  - Record the tuple (sid, C, R, m), and send $(\textsc{Receipt}, \mathsf{sid}, C, R)$ to R and $\mathcal{S}$.
  - Ignore any subsequent $\textsc{Commit}$ command.
- Upon receiving $(\textsc{Decommit}, \mathsf{sid}, C, R)$ from C, do:
  - If there is a tuple (sid, C, R, m) recorded, send $(\textsc{Decommit}, \mathsf{sid}, C, R, m)$ to R and $\mathcal{S}$, and halt.
  - Otherwise, ignore the message.
- When asked by the adversary $\mathcal{S}$, obtain from $\mathcal{G}_{\mathsf{oRO}}$ the list of illegitimate queries $\mathcal{Q}_{\mathsf{sid}}$ that pertain to SID sid, and send $\mathcal{Q}_{\mathsf{sid}}$ to the adversary $\mathcal{S}$.

Figure 6: The Transferable Functionality $\mathcal{F}_{\mathsf{tCOM}}$ for Commitment

We prove this impossibility by contradiction. Suppose that there exists such a 2-round GUC-secure protocol. Let us first consider the case where the receiver is corrupted, the simulator needs to produce an equivocal commitment without knowing the plaintext in the committing phase, and later open it to any given message (a.k.a. the plaintext) in the opening phase. We observe that the receiver does not need to send any message during the 2-round protocol execution, thus when the receiver is controlled by adversary, the corrupted receiver can delay all its $\mathcal{G}_{\mathsf{oRO}}$ queries until it receives the opening message. In this case, the simulator cannot obtain the illegitimate queries of the corrupted receiver before producing the equivocal commitments, and thus has no advantage over the real world adversary. If the simulator still succeeds to produce the equivocal commitments even if it has no illegitimate queries, then distinctions will be revealed when the adversary performs the following attacks. The adversary corrupts the committer, and instructs the committer to run the simulator algorithm mentioned above to generate the commitment message. In this case, where the committer is corrupted, the receiver/simulator needs to extract the plaintext from this commitment message. However, the entire computation of the commitment message is totally independent of the plaintext, thus with high probability the simulation would fail. Formally, we prove this impossibility through Theorem 1.

Theorem 1. *There exists no terminating 2-round (one round for commitment phase and one round for decommitment phase) protocol* Π *that GUC-realizes* $\mathcal{F}_{\mathsf{tCOM}}$ *depicted in Figure 6 with static security, using only the shared functionality for global observable random oracle* $\mathcal{G}_{\mathsf{oRO}}$*.*

Proof. Suppose there exists such a protocol Π that GUC-realizes $\mathcal{F}_{\mathsf{tCOM}}$ in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world. Then there must exist a PPT simulator $\mathcal{S}$ such that $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\mathcal{F}_{\mathsf{tCOM}},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\Pi,\mathcal{A},\mathcal{Z}}$ for any PPT adversary $\mathcal{A}$ and any PPT $\mathcal{G}_{\mathsf{oRO}}$-externally constrained environment $\mathcal{Z}$.

In particular, let us first consider the session with SID sid_1, and let $\mathcal{A}$ be a dummy adversary that simply forwards protocol flows between corrupt parties and the environment $\mathcal{Z}$. Let $\mathcal{Z}$ corrupt the receiver R^* at first. Then $\mathcal{Z}$ chooses a random bit $b \in \{0,1\}$ and gives it as the input to the honest committer C. After that, $\mathcal{Z}$ waits for C to send the commitment ψ. Next, $\mathcal{Z}$ let C reveal the committed value b'. If $b = b'$, $\mathcal{Z}$ outputs 1; otherwise, $\mathcal{Z}$ outputs 0.

In order to make the GUC experiments above remain indistinguishable, the simulator $\mathcal{S}$ needs to build an equivocal commitment ψ without knowing b in the committing phase; later in the opening phase, $\mathcal{S}$ obtains b from $\mathcal{F}_{\mathsf{tCOM}}$ and needs to open the previously sent commitment ψ to b. Recall that, the main advantage of $\mathcal{S}$ over the others is that it can obtain illegitimate queries of R^*. More precisely, $\mathcal{S}$ can request the illegitimate queries $\mathcal{Q}_{\mathsf{sid}_1}$ from the commitment functionality $\mathcal{F}_{\mathsf{tCOM}}$ who forwards this request to $\mathcal{G}_{\mathsf{oRO}}$. The simulator $\mathcal{S}$ also can query $\mathcal{G}_{\mathsf{oRO}}$ just like normal parties, and we denote by $\tilde{\mathcal{Q}}_{\mathsf{sid}_1,\mathcal{S}}$ the queries sent by $\mathcal{S}$ in the committing phase. With these notions, we write $\psi \leftarrow \mathsf{FakeCom}(\mathsf{sid}_1, \tilde{\mathcal{Q}}_{\mathsf{sid}_1,\mathcal{S}}, \mathcal{Q}_{\mathsf{sid}_1})$ to denote the event where $\mathcal{S}$ produces the equivocal commitment ψ using $\mathcal{Q}_{\mathsf{sid}_1}$ and $\tilde{\mathcal{Q}}_{\mathsf{sid}_1,\mathcal{S}}$. We note that, $\mathcal{S}$ should be able to handle any PPT environment $\mathcal{Z}$. Consider such a case where $\mathcal{Z}$ instruct R^* to delay all its $\mathcal{G}_{\mathsf{oRO}}$ queries until it receives the opening message. In this case, $\mathcal{S}$ may find nothing useful in $\mathcal{Q}_{\mathsf{sid}_1}$, but should still be able to produce the equivocal commitment ψ which we can write as $\psi \leftarrow \mathsf{FakeCom}(\mathsf{sid}_1, \tilde{\mathcal{Q}}_{\mathsf{sid}_1,\mathcal{S}}, \emptyset)$, where $\emptyset$ is an empty set. We also note that, there is nothing related to b in the queries $\tilde{\mathcal{Q}}_{\mathsf{sid}_1,\mathcal{S}}$ since b is the hidden input from $\mathcal{S}$ theoretically. Similarly, if we denote by $\bar{\mathcal{Q}}_{\mathsf{sid}_1,\mathcal{S}}$ the queries sent by $\mathcal{S}$ in the opening phase, then we can write $r \leftarrow \mathsf{FakeOpen}(\mathsf{sid}_1, \psi, b, \bar{\mathcal{Q}}_{\mathsf{sid}_1,\mathcal{S}}, \emptyset)$ to denote the event where $\mathcal{S}$ is provided with useless $\mathcal{Q}_{\mathsf{sid}_1}$, but should still be able to to open ψ to the value b and the corresponding opening message r using $\bar{\mathcal{Q}}_{\mathsf{sid}_1,\mathcal{S}}$. We can regard $\mathsf{FakeCom}, \mathsf{FakeOpen}$ as PPT algorithms. If we switch to the session with a different SID, these algorithms still work as long as we provide the appropriate input.

In the following, we show that the existence of the simulator $\mathcal{S}$ above contradicts the security of Π against static corruptions, by creating a particular environment $\mathcal{Z}'$ which succeeds in distinguishing $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\mathcal{F}_{\mathsf{tCOM}},\mathcal{S}',\mathcal{Z}'}$ from $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\Pi,\mathcal{A}',\mathcal{Z}'}$ after a static corruption operation for any PPT simulator $\mathcal{S}'$. Let us consider the session with SID sid_2. Our $\mathcal{Z}'$ proceeds by corrupting the committer C^* at first, and then choosing a random bit $b \in \{0,1\}$ which it gives as the input to C^*. Next $\mathcal{Z}'$ instructs C^* to run the algorithm $\psi \leftarrow \mathsf{FakeCom}(\mathsf{sid}_2, \tilde{\mathcal{Q}}_{\mathsf{sid}_2,\mathcal{S}}, \emptyset)$ and send ψ to R. When R outputs $(\textsc{Receipt}, \mathsf{sid}_2, C, R)$, $\mathcal{Z}'$ instructs C^* to run $r \leftarrow \mathsf{FakeOpen}(\mathsf{sid}_2, \psi, b, \bar{\mathcal{Q}}_{\mathsf{sid}_2,\mathcal{S}}, \emptyset)$ and send (b, r) to R. Finally $\mathcal{Z}'$ waits for R to output b'. In the real world, R always outputs $b' = b$. In the ideal world, $\mathcal{S}'$ should determine the committed value b' from ψ in the committing phase. This means that in the ideal world, we must have that $b' = b$ with probability at most $\frac{1}{2}$, since the entire computation of ψ is totally independent of b. Therefore, $\mathcal{Z}'$ can distinguish between the real world and the ideal world with probability at least $\frac{1}{2}$, contradicting our assumption that Π is GUC-secure. $\square$

4 Feasibility in the GORO Model

In this section, we propose a 3-round (2 rounds for the committing phase and 1 round for the opening phase) GUC-secure commitment protocol in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world, assuming the straight-line extractable NIWH arguments and perfect-hiding non-interactive equivocal commitment schemes exist. Then we instantiate the building blocks using only MiniCrypt assumptions in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world. Therefore, our GUC-secure commitment protocol can be constructed via MiniCrypt in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world. Since we prove that it is impossible to construct 2-round GUC-secure commitments in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world in Theorem 1, we stress that our construction is round-optimal.

4.1 Our GUC-Secure Commitment Construction

Recall that a GUC-secure commitment protocol requires two main properties: (i) *Equivocality:* when the receiver is corrupted, the simulator should be able to produce equivocal commitments that can open to any value later; (ii) *Extractability:* when the committer is corrupted, the simulator should be able to extract the committed value from the commitment.

The $\mathcal{G}_{\mathsf{oRO}}$ directly provides the desired extractability. Then we have to design a protocol that capture the equivocality. A natural approach is to employ the perfect-hiding non-interactive equivocal commitments. More precisely, we let the receiver generate the commitment key and send it to the committer in the first round; and then let the committer commit to the message using the equivocal commitment scheme. In order to provide extractability, we let the committer query the $\mathcal{G}_{\mathsf{oRO}}$ on the opening message of the commitment message above. Then we require the committer to commit to the answer of the $\mathcal{G}_{\mathsf{oRO}}$ via another instance of the equivocal commitment scheme. The committer sends all the commitment messages in the second round. The opening phase just takes one round, namely, the committer sends all the opening messages.

Protocol Π_{tCOM}

Primitives: Straight-line extractable NIWH argument $\Pi_{\mathsf{NIWH}} := (P, V)$, non-interactive equivocal commitment $\mathsf{ECom} := (\mathsf{KeyGen}, \mathsf{KeyVer}, \mathsf{Commit}, \mathsf{ComVer}, \mathsf{EquCom}, \mathsf{Equiv})$.
Inputs: C has a private input $m \in \{0,1\}^\lambda$. R has no input.

Committing Phase: This phase consists of 2 rounds.

- Round 1: R works as follows:
  - Generate the commitment key and the equivocation trapdoor by invoking $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
  - Compute the NIWH proof by invoking $\pi \leftarrow \Pi_{\mathsf{NIWH}}.P(\mathsf{ck}, \mathsf{td})$ for proving the knowledge of td. Send (ck, π) to C.
- Round 2: C works as follows:
  - Check the validity of π by invoking $b \leftarrow \Pi_{\mathsf{NIWH}}.V(\mathsf{ck}, \pi)$, and abort if $b = 1$
  - Commit to the message m by invoking $(c_1, d_1) \leftarrow \mathsf{ECom.Commit}(\mathsf{ck}, m)$.
  - Compute $h \leftarrow \mathsf{oRO}(\mathsf{sid}, `C'||m||d_1||r)$ where r is a λ-bit randomness.
  - Commit to the answer h by invoking $(c_2, d_2) \leftarrow \mathsf{ECom.Commit}(\mathsf{ck}, h)$.
  - Send (c_1, c_2) to R.

Opening Phase: This phase consists of 1 round.

- Round 3: C sends (m, d_1, d_2, r) to R.
- R computes $h \leftarrow \mathsf{oRO}(\mathsf{sid}, `C'||m||d_1||r)$, and accepts m if and only if $\mathsf{ECom.ComVer}(\mathsf{ck}, c_1, d_1) = \mathsf{ECom.ComVer}(\mathsf{ck}, c_2, d_2) = 1$ holds.

Figure 7: Protocol Π_{tCOM} in the $\mathcal{G}_{\mathsf{oRO}}$ Hybrid World

The only thing left is to provide the simulator with the advantage of getting the equivocation trapdoor over the others. Our solution is to let the receiver execute the straight-line extractable NIWH argument which proves the knowledge of the equivocation trapdoor with respect to the commitment key. The receiver is required to send the proof along with the commitment key in the first round. Subsequently, the simulator can invoke the NIWH straight-line extractor to obtain the equivocation trapdoor.

We denote committer algorithm as C and receiver algorithm as R. We denote the event where queries $\mathcal{G}_{\mathsf{oRO}}$ on input x and gets the answer y as $y \leftarrow \mathsf{oRO}(x)$ in the context. We assume ideal private and authenticated channels for all communications. Formally, we present our protocol Π_{tCOM} in Figure 7 and prove the security through Theorem 2.

Theorem 2. *Assume Π_{NIWH} is a NIWH argument with a straight-line extractor. Assume* ECom *is an equivocal commitment scheme. Then the protocol Π_{tCOM} described in Figure 7 GUC-realizes the functionality $\mathcal{F}_{\mathsf{tCOM}}$ depicted in Figure 6 in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world against static malicious corruption.*

Remark 2. *We leave the proof in Appendix A.*

4.2 Instantiation of the Building Blocks

There are two building blocks, i.e. straight-line extractable NIWH arguments and perfect-hiding non-interactive equivocal commitment schemes, in our construction. In this section, we show how to instantiate them using only MiniCrypt assumptions in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world. We start by constructing a SHVZK protocol, since it is needed in both building blocks.

4.2.1 SHVZK Protocols from "MPC-in-the-Head"

In this section, we aim to construct a SHVZK protocol using only MiniCrypt assumptions in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world. Our starting point is the "MPC-in-the-head" paradigm proposed by Ishai *et al.* [IKOS07]. Let $f_{\mathcal{R}}$ be the following $(n+1)$-argument function corresponding to an NP relation $\mathcal{R}$: $f_{\mathcal{R}}(x, w_1, \cdots, w_n) = \mathcal{R}(x, w_1 \oplus \cdots \oplus w_n)$. Here $f_{\mathcal{R}}$ can be viewed as an n-party functionality, where x is a public input known to all parties, w_i is the private input of party P_i, and the output is received by all parties. In a high-level description, the main idea is for the prover to simulate the execution of a t-private n-party MPC protocol that realizes $f_{\mathcal{R}}$. Then the prover employs a statically binding commitment to commit to all views of the parties and sends them to the verifier. After that, the verifier chooses a random subset of the parties, where the size of the subset equals t, and asks the prover to open their corresponding views. Finally the verifier checks that the received views are consistent with each other.

Note that our construction requires an SHVZK protocol with 2-special soundness (, which we will explain the necessarity later in Section 4.2.2, below); unfortunately, to the best of our knowledge, none of the followups [GMO16, CDG$^+$17, AHIV17, KKW18, dOT21] since the original work of [IKOS07], can lead to a 2-special sound protocol based on only MiniCrypt assumptions. Therefore, we need to design a new technique approach that transforms a MPC protocol into a SHVZK protocol with 2-special soundness.

We start with the 5-round SHVZK protocol proposed by Katz *et al.* in [KKW18] which is based on only MiniCrypt assumptions. In the high-level description, Katz *et al.* employed the $(n-1)$-private n-player MPC protocol in the preprocessing model and let the verifier provide its challenges in *two* phases: one for checking the correctness of the opened preprocessing execution, and the other for checking the consistency of the opened views. Roughly speaking, the 5-round protocol of Katz *et al.* works as follows:

- Round 1: The prover simulates m independent executions of the preprocessing phase, and commits to the states of the parties which can be obtained at the end of the preprocessing phase.
- Round 2: The verifier samples an uniform random challenge $c \in [m]$ and asks the prover to open the views of all the executions of the preprocessing phase except the c-th one.
- Round 3: The prover opens the states of all parties for each challenged execution of preprocessing phase. Beside that, the prover simulates the execution of Π_{MPC} that realizes $f_{\mathcal{R}}$ using the remaining unopened execution of the preprocessing phase. The prover then commits to each view of the parties.
- Round 4: The verifier samples an uniform random challenge $p \in [n]$ and asks the prover to open all the views of the parties except the p-th one.
- Round 5: The prover reveals the state of each challenged party following the preprocessing phase as well as its view in the execution of Π_{MPC}. The verifier checks that the opened views are consistent with each other and with an honest execution of Π_{MPC} (using the state from the preprocessing phase) that yields the output 1.

In [KKW18], Katz *et al.* compressed the above 5-round protocol into a 3-round one by the following approach: (i) let the prover simulate the execution of Π_{MPC} for every emulation of the preprocessing phase and commit to all the resulting views as well as the states; (ii) let the verifier send its challenge (c, p), and asks the prover to open all the states except the c-th one of the preprocessing phase as well as all the views except the p-th one from the unopened preprocessing phase. We emphasize that the resulting 3-round protocol is not 2-special sound, since the adversary can produce two distinct accepting transcripts without knowing the witness as follows: it first sets challenges as $(c, p), (c', p)$ where $c \neq c'$, then emulates all the preprocessing phase honestly. Finally, it runs the simulator of Π_{MPC} to prepare two sets of the opened views, samples the random strings as the unopened views, and commits to all the views. Formally, we have the following proposition, and leave the formal proof in Appendix D.

Proposition 1. *Assume the n-party MPC protocol Π_{MPC} is $(n-1)$-private in the preprocessing model. Let m be the number of executions of preprocessing phase, where $m \geq 2$. The 3-round SHVZK protocol described in [KKW18]:*

- *cannot achieve k-special soundness, for $k \leq m$.*
- *satisfies k-special soundness, for $k \geq m+1$.*

Our key observation is that we can compress the above 5-round protocol into a 3-round one by applying the Fiat-Shamir transformation [FS87] to replace Round 2. Therefore, Round 1 and Round 3 can be merged, and we obtain a 3-round protocol with 2-special soundness. We can regard the first round of the resulting 3-round protocol as a "non-interactive proof" that proves the correctness of the execution of the preprocessing phase, but its soundness error is not negligible (i.e., $\frac{1}{m}$, where m is the number of the executions of preprocessing phase). This issue can be addressed by applying parallel repetition. Compared with the approach of Katz *et al.*, our approach needs additional RO assumptions but it is an SHVZK protocol with 2-special sound.

Given the MPC protocol Π_{MPC} which realizes $f_{\mathcal{R}}$ with perfect $(n-1)$-privacy in the preprocessing model, we can obtain a SHVZK protocol from the paradigm mentioned in the previous paragraph, using the $\mathcal{G}_{\mathsf{oRO}}$ to instantiate the statically binding commitment (i.e., to commit msg with random coin r, we use the answer of the $\mathcal{G}_{\mathsf{oRO}}$ on input (msg, r) as the commitment and reveal (msg, r) as the opening). We denote the event where queries $\mathcal{G}_{\mathsf{oRO}_i}$ on input x and gets the answer y as $y \leftarrow \mathsf{oRO}_i(x)$ for $i \in \{1, 2\}$ in the context, where $\mathsf{oRO}_1 : \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} \to \{0,1\}^{\ell}$ and $\mathsf{oRO}_2 : \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} \to (\mathbb{Z}^{+}_{m+1})^{\lambda}$. We denote by m the number of the executions of the preprocessing phase. Formally, we present our protocol Π_{SHVZK} in Figure 8 and prove the security through Theorem 3.

Theorem 3. *Assume Π_{MPC} is a secure n-party protocol that realizes $f_{\mathcal{R}}$ with perfect $(n-1)$-privacy, where $f_{\mathcal{R}}(x, w_1, \cdots, w_n) = \mathcal{R}(x, w_1 \oplus \cdots \oplus w_n)$. Then the protocol Π_{SHVZK} depicted in Figure 8 is a SHVZK protocol that is perfect complete, 2-special sound, perfect SHVZK.*

Proof. Perfect Completeness. It is straightforward.

Perfect SHVZK. We have already presented the strategy of the simulator in Figure 8. The only thing left is to show the indistinguishability. Since the simulator emulates the executions of the preprocessing phase honestly, the real states of the parties and the simulated ones are perfectly indistinguishable. Now we turn to the views of the parties: given $e := (p_1, \ldots, p_\lambda)$ ahead, by perfect $(n-1)$-privacy of Π_{MPC}, the real opened views of the parties and simulated ones are perfectly indistinguishable. Thus, the real z and simulated z' are perfectly indistinguishable. Due to the unpredictability of the output of $\mathcal{G}_{\mathsf{oRO}}$, the real a and simulated a' are perfectly indistinguishable. Therefore, we prove that our protocol satisfies perfect SHVZK property.

2-Special Soundness. Fixing $a := (\{\mathsf{com}_{i,j,k}, \widetilde{\mathsf{com}}_{i,k}\}_{i\in[\lambda], j\in[m], k\in[n]}, \{\mathsf{state}_{i,j,k}, r_{i,j,k}\}_{i\in[\lambda], j\in[m]/c_i, k\in[n]})$, for any two distinct accepting transcripts $(a, e := (p_1, \ldots, p_\lambda), z), (a, e' := (p'_1, \ldots, p'_\lambda), z')$ where $e \neq e'$, we can find a $i \in [\lambda]$ such that $p_i \neq p'_i$. Thus, we can obtain all the committed views $\{\mathsf{view}_{i,k}(x, w_k), \mathsf{state}_{i,c_i,k}\}_{k\in[n]}$ given these transcripts. Then the extractor $\Pi_{\mathsf{SHVZK}}.\mathsf{Ext}$ yields w_k from $\mathsf{view}_i(x, w_k), \mathsf{state}_{i,c_i,k}$ and computes $w \leftarrow w_1 \oplus \cdots \oplus w_n$. Since all the views and states has been verified due to the definition of the special soundness, we argue that the probability of $\Pi_{\mathsf{SHVZK}}.\mathsf{Ext}$ failing to output a valid w such that $(x, w) \in \mathcal{R}$ is 0. Therefore, we prove that our protocol is 2-special sound. □

Protocol Π_{SHVZK}

Primitives: n-party MPC protocol Π_{MPC} which realizes $f_{\mathcal{R}}$ with $(n-1)$-privacy in the preprocessing model, where $f_{\mathcal{R}}(x, w_1, \cdots, w_n) = \mathcal{R}(x, w_1 \oplus \cdots \oplus w_n)$.
Random Oracles: $\mathsf{oRO}_1 : \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} \to \{0,1\}^{\ell}$ and $\mathsf{oRO}_2 : \{0,1\}^{\ell_{\mathsf{in}}(\lambda)} \to (\mathbb{Z}_{m+1}^{+})^{\lambda}$
Inputs: P, V both have a public input x and an NP relation $\mathcal{R}$. P has a private input w such that $\mathcal{R}(x, w) = 1$.

Protocol:

- $\mathcal{P}_1(x, w, r)$:
  - For $i \in [\lambda], j \in [m]$:
    - Derive λ-bit random $\mathsf{seed}_{i,j}$ from randomness r and generate $\{\mathsf{state}_{i,j,k}\}_{k\in[n]} \leftarrow \mathsf{Preprocess}(\mathsf{seed}_{i,j})$.
    - For $k \in [n]$: select $r_{i,j,k} \xleftarrow{\$} \{0,1\}^{\lambda}$ and commit to the states $\mathsf{com}_{i,j,k} \leftarrow \mathsf{oRO}_1(\mathsf{state}_{i,j,k}, r_{i,j,k})$.
  - Compute $(c_1, \ldots, c_{\lambda}) \leftarrow \mathsf{oRO}_2(\{\mathsf{com}_{i,j,k}\}_{i\in[\lambda],j\in[m],k\in[n]})$, where $\forall i \in [\lambda] : c_i \in [m]$.
  - For $i \in [\lambda]$:
    - Simulate the execution of Π_{MPC} using (x, w) and the states generated by the c_i-th preprocessing phase (i.e., $\{\mathsf{state}_{i,c_i,k}\}_{k\in[n]}$), and output the views of the parties $\{\mathsf{view}_{i,k}(x, w_k)\}_{k\in[n]}$.
    - For $k \in [n]$: select $\tilde{r}_{i,k} \xleftarrow{\$} \{0,1\}^{\lambda}$ and commit to the view of each party $\widetilde{\mathsf{com}}_{i,k} \leftarrow \mathsf{oRO}_1(\mathsf{view}_{i,k}(x, w_k), \tilde{r}_{i,k})$.
  - Send $a := (\{\mathsf{com}_{i,j,k}, \widetilde{\mathsf{com}}_{i,k}\}_{i\in[\lambda],j\in[m],k\in[n]}, \{\mathsf{state}_{i,j,k}, r_{i,j,k}\}_{i\in[\lambda],j\in[m]/c_i,k\in[n]})$
- $\mathcal{V}_1(1^{\lambda})$: Send $e := (p_1, \ldots, p_{\lambda})$, where $p_i \in [n]$ and p_i is uniformly random.
- $\mathcal{P}_2(x, w, r, e)$: Send $z := (\{\mathsf{view}_{i,k}(x, w_k), \tilde{r}_{i,k}, \mathsf{state}_{i,c_i,k}, r_{i,c_i,k}\}_{i\in[\lambda],k\in[n]/p_i})$.
- $\mathcal{V}_2(x, a, e, z)$: Output 1 if and only if the following checks pass:
  - Check the commitments are opened correctly:
    - For $i \in [\lambda], j \in [m]/c_i, k \in [n]$: check $\mathsf{com}_{i,j,k} = \mathsf{oRO}_1(\mathsf{state}_{i,j,k}, r_{i,j,k})$ holds.
    - For $i \in [\lambda], k \in [n]/p_i$: check $\mathsf{com}_{i,c_i,k} = \mathsf{oRO}_1(\mathsf{state}_{i,c_i,k}, r_{i,c_i,k})$ and $\widetilde{\mathsf{com}}_{i,k} = \mathsf{oRO}_1(\mathsf{view}_{i,k}(x, w_k), \tilde{r}_{i,k})$ hold.
  - Check the correctness of the executions of the preprocessing phase:
    - Compute $(c_1, \ldots, c_{\lambda}) \leftarrow \mathsf{oRO}_2(\{\mathsf{com}_{i,j,k}\}_{i\in[\lambda],j\in[m],k\in[n]})$.
    - For $i \in [\lambda], j \in [m]/c_i$: check $\{\mathsf{state}_{i,j,k}\}_{k\in[n]}$ are well-formed.
  - Check the consistency between the opened views:
    - For $i \in [\lambda], k \in [n]/p_i$: check $\mathsf{view}_{i,k}(x, w_k)$ follows from the $\mathsf{state}_{i,c_i,k}$ correctly and $\mathsf{view}_{i,k}(x, w_k)$ yields output 1.
    - For $i \in [\lambda]$: check $\{\mathsf{view}_{i,k}(x, w_k)\}_{k\in[n]/p_i}$ are consistent with each other.

Simulation: Given the challenge $e := (p_1, \ldots, p_{\lambda})$ ahead, we have

- $\mathsf{Sim}(x, e, r)$:
  - For $i \in [\lambda], j \in [m]$:
    - Derive λ-bit random $\mathsf{seed}_{i,j}$ from randomness r and generate $\{\mathsf{state}_{i,j,k}\}_{k\in[n]} \leftarrow \mathsf{Preprocess}(\mathsf{seed}_{i,j})$.
    - For $k \in [n]$: select $r_{i,j,k} \xleftarrow{\$} \{0,1\}^{\lambda}$ and commit to the states $\mathsf{com}_{i,j,k} \leftarrow \mathsf{oRO}_1(\mathsf{state}_{i,j,k}, r_{i,j,k})$.
  - Compute $(c_1, \ldots, c_{\lambda}) \leftarrow \mathsf{oRO}_2(\{\mathsf{com}_{i,j,k}\}_{i\in[\lambda],j\in[m],k\in[n]})$, where $\forall i \in [\lambda] : c_i \in [m]$.
  - For $i \in [\lambda]$:
    - Run the simulator algorithm of Π_{MPC} using x, p_i and the states generated by the c_i-th preprocessing phase (i.e., $\{\mathsf{state}_{i,c_i,k}\}_{k\in[n]}$), and output the simulated views $\{\mathsf{view}_{i,k}\}_{k\in[n]/p_i}$. Sample a random view_{i,p_i} of appropriate length.
    - For $k \in [n]$: select $\tilde{r}_{i,k} \xleftarrow{\$} \{0,1\}^{\lambda}$ and commit to the view of each party $\widetilde{\mathsf{com}}_{i,k} \leftarrow \mathsf{oRO}_1(\mathsf{view}_{i,k}, \tilde{r}_{i,k})$.
  - Output $a := (\{\mathsf{com}_{i,j,k}, \widetilde{\mathsf{com}}_{i,k}\}_{i\in[\lambda],j\in[m],k\in[n]}, \{\mathsf{state}_{i,j,k}, r_{i,j,k}\}_{i\in[\lambda],j\in[m]/c_i,k\in[n]})$ and $z := (\{\mathsf{view}_{i,k}, \tilde{r}_{i,k}, \mathsf{state}_{i,c_i,k}, r_{i,c_i,k}\}_{i\in[\lambda],k\in[n]/p_i})$.

Figure 8: Protocol Π_{SHVZK} in the $\mathcal{G}_{\mathsf{oRO}}$ Hybrid World

4.2.2 Perfect-Hiding Non-Interactive Equivocal Commitment

Given a SHVZK protocol, we can obtain a perfect-hiding non-interactive equivocal commitment. The intuition is as follows. Let $\mathcal{R}$ be a hard NP relation. The receiver selects $(x, w) \in \mathcal{R}$, and sets x as the commitment key and w as the equivocation trapdoor. The message m is used as the challenge on which to run the simulator for the SHVZK-PoK protocol with respect to x, producing the prover's first flow a and the response z. The first flow a is used as the commitment. The message m and response z are used as the opening. Equivocation is achieved by using the knowledge of w to execute the honest prover algorithm instead of the simulator algorithm. Similar ideas can be found in [Dam02, MY04].

Let OWF be a one-way function. Formally, we present our non-interactive equivocal commitment in Figure 9 and prove the security through Theorem 4. The proof of computational binding relies on the 2-special soundness, and this explains the reason why 2-special soundness is necessary in Section 4.2.1. Note that, we instantiate the NP relation with one-way function $\mathcal{R} := \{(y, \mathsf{seed}) \mid y = \mathsf{OWF}(\mathsf{seed})\}$, where y is the statement and seed is the witness. If we use our SHVZK protocol Π_{SHVZK} depicted in Figure 8 as the building block, then we can obtain a perfect hiding non-interactive equivocal commitment via only MiniCrypt assumptions in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world.

Construction ECom

Primitives: SHVZK protocol $\Pi_{\mathsf{SHVZK}} := (\mathcal{P}_1, \mathcal{P}_2, \mathcal{V}_1, \mathcal{V}_2)$ and one-way function OWF.

- $\mathsf{KeyGen}(1^\lambda)$: Select a λ-bit random string seed, compute $y \leftarrow \mathsf{OWF}(\mathsf{seed})$, and output $\mathsf{ck} := y, \mathsf{td} := \mathsf{seed}$.
- $\mathsf{KeyVer}(\mathsf{ck}, \mathsf{td})$: Check if $y = \mathsf{OWF}(\mathsf{seed})$ holds. If so, output 1; otherwise, output 0.
- $\mathsf{Commit}(\mathsf{ck}, m)$: Select a λ-bit randomness r, invoke $(a, z) \leftarrow \Pi_{\mathsf{SHVZK}}.\mathsf{Sim}(\mathsf{ck}, m, r)$, and output $c := a, d := (m, z)$.
- $\mathsf{ComVer}(\mathsf{ck}, c, d)$: Check if $\Pi_{\mathsf{SHVZK}}.\mathcal{V}_2(\mathsf{ck}, c, m, z) = 1$ holds. If so, output 1; otherwise, output 0.
- $\mathsf{EquCom}(\mathsf{ck}, \mathsf{td})$: Select a λ-bit randomness r, invoke $\tilde{a} \leftarrow \Pi_{\mathsf{SHVZK}}.\mathcal{P}_1(y, \mathsf{seed}, r)$, and output $\tilde{c} := \tilde{a}, \mathsf{st} := r$.
- $\mathsf{Equiv}(\mathsf{ck}, \mathsf{td}, \tilde{c}, \mathsf{st}, \tilde{m})$: Invoke $\tilde{z} \leftarrow \Pi_{\mathsf{SHVZK}}.\mathcal{P}_2(y, \mathsf{seed}, r, \tilde{m})$, and output $\tilde{d} := \tilde{z}$.

Figure 9: Construction ECom Based on One-Way Function

Theorem 4. *Assume Π_{SHVZK} is a SHVZK protocol that is 2-special sound and perfect SHVZK. Assume OWF is a one-way function that is hard to invert. Then ECom depicted in Figure 9 is an equivocal commitment that is perfect correct, perfect hiding, computational binding and perfect equivocal.*

Proof. Perfect Correctness. It is straightforward.

Perfect Hiding. We proceed by contradiction. Assume there exists an unbounded adversary $\mathcal{A}$ that breaks the perfect hiding property of the ECom with non-negligible probability, then we are able to build a reduction $\mathcal{B}$ which violates the perfect SHVZK property of the underlying Π_{SHVZK}. First $\mathcal{B}$ interacts with two SHVZK game challengers $\mathcal{C}_0, \mathcal{C}_1$. Then $\mathcal{B}$ starts the protocol with $\mathcal{A}$ by running $\mathcal{A}$ internally as a black-box. Next, $\mathcal{B}$ selects a random seed, computes $y \leftarrow \mathsf{OWF}(\mathsf{seed})$ and sends y to $\mathcal{A}$. After receiving two distinct messages m_0, m_1 from $\mathcal{A}$, $\mathcal{B}$ selects a random bit $b \in \{0, 1\}$ and sends (y, seed, m_b) to $\mathcal{C}_b$. Upon receiving c from $\mathcal{C}_b$, $\mathcal{B}$ forwards the message to $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ simply forwards b' to $\mathcal{C}_b$. Note that, the probability of c is produced by the SHVZK simulator is $\frac{1}{2}$, thus we have $\Pr[\mathcal{B} \text{ wins}] = \frac{1}{2}\Pr[\mathcal{A} \text{ wins}]$. Therefore, we prove that our construction is perfect hiding.

Computational Binding. We proceed by contradiction. Assume there exists a PPT adversary $\mathcal{A}$ that breaks the computational binding property of the ECom with non-negligible probability, then we are able to build a reduction $\mathcal{B}$ which violates the Hard to Invert (HI) property of the underlying OWF. First $\mathcal{B}$ interacts with HI game challenger $\mathcal{C}$. After receiving y from $\mathcal{C}$, $\mathcal{C}$ forwards y to $\mathcal{A}$. Then $\mathcal{B}$ waits for $\mathcal{A}$ to send $c := a, d := (m_1, z_1), d' := (m_2, z_2)$, where $m_1 \neq m_2$. Due to the 2-special soundness, $\mathcal{B}$ is able to extract the witness seed such that $y = \mathsf{OWF}(\mathsf{seed})$, sends seed to $\mathcal{C}$ and wins the game. Therefore, we prove that our construction is computational binding.

Perfect Equivocal. In order to prove perfect equivocal property, we have to show that the real commitment-opening pair $(c := a, d := (m, z))$ is perfectly indistinguishable from the equivocated commitment-opening pair $(\tilde{c} := \tilde{a}, \tilde{d} := (m, \tilde{z}))$. By perfect SHVZK property of Π_{SHVZK}, (a, z) is perfectly indistinguishable from $(\tilde{a}, \tilde{z})$. Therefore, we prove that our construction is perfect equivocal. $\square$

4.2.3 Straight-Line Extractable NIWH Argument

We construct the straight-line extractable NIWH argument using the technique described in [Pas03]. We here describe the high-level description and the details can be found in Appendix E. Given a SHVZK protocol with 2-special soundness, we let the prover execute the honest prover algorithm to obtain the first flow message. Fixing this first flow message, we let the prover pick two distinct random challenges and compute the corresponding responses. Then the prover commits to the response by querying $\mathcal{G}_{\mathsf{oRO}}$ and using the answer as the commitment. Next the prover sends the first flow message along with all the challenges and the commitments to the verifier. After that, the verifier asks the prover to open one commitment. Finally the verifier receives the response, and checks if the corresponding transcript is valid. The soundness error of the protocol described above is $\frac{1}{2}$, and it can be reduced by parallel repetitions. We also apply Fiat-Shamir transformation to remove the interaction [FS87]. The straight-line extractablity relies on the observability provided by $\mathcal{G}_{\mathsf{oRO}}$ and 2-special soundness.

Theorem 5 ([Pas03])**.** *Assume there is a 2-special sound SHVZK protocol, then there exists a straight-line extractable NIWH argument in the $\mathcal{G}_{\mathsf{oRO}}$ model.*

If we instantiate the 2-special sound SHVZK protocol with ours that depicted in Figure 8, then the resulting straight-line extractable NIWH argument also only requires MiniCrypt assumptions. We show a more general case that transforms a k-special sound SHVZK protocol for $k \geq 2$, into a straight-line NIWH argument in Appendix E.

5 Concluding Remarks: Towards a Complete Picture

In this work, we mainly focus on the lower bounds on round complexity for GUC-secure commitment protocols in the global random oracle models. We also wonder if such lower bounds exist, is it possible to construct round-optimal GUC-secure commitment protocols via MiniCrypt assumptions?

In terms of the $\mathcal{G}_{\mathsf{oRO}}$, our work gives a complete answer: we show it is impossible to construct 2-round GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world against static adversaries in Section 3, and construct a 3-round (round-optimal) GUC-secure commitment protocol via MiniCrypt in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world in Section 4. In the remaining, let us turn our attention on other global random oracle models.

As for the $\mathcal{G}_{\mathsf{sRO}}$, the results of [CDPW07] rules out the possibility of constructing any GUC-secure commitment protocol in the $\mathcal{G}_{\mathsf{sRO}}$ hybrid world. More precisely, they argued that no "public setup", namely no setup that provides only public information that is available to all parties, can suffice for realizing commitment protocols in the GUC framework. It is easy to see that this impossibility result holds in the $\mathcal{G}_{\mathsf{sRO}}$ hybrid world.

Regarding the $\mathcal{G}_{\mathsf{poRO}}$, non-interactive GUC-secure commitment protocol can be achieved. In fact, Camenisch *et al.* proposed a non-interactive GUC-secure commitment in the $\mathcal{G}_{\mathsf{poRO}}$ hybrid world without any further assumptions [CDG+18].

Among all the global random oracle models depicted in Figure 3, only the $\mathcal{G}_{\mathsf{pRO}}$ has yet to be fully investigated. Actually, we already have some impossibility result: we find that there exists *no* GUC-secure commitment protocols with one-round committing phase in the $\mathcal{G}_{\mathsf{pRO}}$ hybrid world against static adversaries. Intuitively, we observe that the receiver does not have the chance to send any message in the committing phase in such commitment protocols. Note that, the global programmable RO only allows the simulator to program on the unqueried points without being detected, and the simulator benefits itself by letting the corrupted parties to work on its programmed points. Now let us consider the case where the committer is corrupted and the simulator acts as the receiver, the simulator needs to extract the committed value before the opening phase. In a commitment protocol where the committing phase only takes one round, the simulator (acting as the receiver) does not need to send any message, thus it cannot enforce the corrupted committer to produce its message on the programmed points; if it could, then we can use such a simulator to break the hiding property of the commitment since anyone can run this simulator without relying on the programmability of the $\mathcal{G}_{\mathsf{pRO}}$. In conclusion, the committing phase requires at least 2 rounds, plus (at least) 1 round of the opening phase, and the entire commitment protocol requires at least 3 rounds. We refer interesting readers to see the formal theorem and proof in Appendix B.

Given this lower bound in the $\mathcal{G}_{\mathsf{pRO}}$, we find the 3-round (2 rounds for the committing phase, 1 round for the opening phase) GUC-secure commitment protocol proposed in [CDG+18] is round-optimal. But their construction relies on CDH assumption which lives in CryptoMania world. The sad truth is that we find it extremely hard to construct a round-optimal GUC-secure commitment protocol via MiniCrypt in the $\mathcal{G}_{\mathsf{pRO}}$ hybrid world, so we leave it as an open question.

References

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017. 3, 16

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046. 1

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th FOCS*, pages 186–195. IEEE Computer Society Press, October 2004. 4

[BDD20] Carsten Baum, Bernardo David, and Rafael Dowsley. Insured MPC: Efficient secure computation with financial penalties. In Joseph Bonneau and Nadia Heninger, editors, *FC 2020*, volume 12059 of *LNCS*, pages 404–420. Springer, Heidelberg, February 2020. 4

[BGM19] Pedro Branco, Manuel Goulão, and Paulo Mateus. UC-commitment schemes with phase-adaptive security from trapdoor functions. Cryptology ePrint Archive, Report 2019/529, 2019. https://eprint.iacr.org/2019/529. 4

[BPRS17] Megha Byali, Arpita Patra, Divya Ravi, and Pratik Sarkar. Fast and universally-composable oblivious transfer and commitment scheme with adaptive security. Cryptology ePrint Archive, Report 2017/1165, 2017. https://eprint.iacr.org/2017/1165. 4

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. 1

[Bra21] Pedro Branco. A post-quantum uc-commitment scheme in the global random oracle model from code-based assumptions. *Advances in Mathematics of Communications*, 15(1):113, 2021. 4

[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, 5

[CDG$^+$17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1825–1842. ACM Press, October / November 2017. 3, 16

[CDG$^+$18] Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Heidelberg, April / May 2018. 1, 4, 6, 8, 20, 21

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, February 2007. 1, 3, 4, 5, 6, 20

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001. 1, 4

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998. 1

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014. 1, 2, 3, 4, 6, 13

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020. 1

[CSW20] Ran Canetti, Pratik Sarkar, and Xiao Wang. Efficient and round-optimal oblivious transfer and commitment with adaptive security. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 277–308. Springer, Heidelberg, December 2020. 1, 4

[Dam02] Ivan Damgård. On σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, page 84, 2002. 3, 19

[dOT21] Cyprien de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge MPCitH-based arguments. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 3022–3036. ACM Press, November 2021. 3, 16

[DSW08] Yevgeniy Dodis, Victor Shoup, and Shabsi Walfish. Efficient constructions of composable commitments and zero-knowledge proofs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 515–535. Springer, Heidelberg, August 2008. 1, 4

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. 17, 20

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *22nd ACM STOC*, pages 416–426. ACM Press, May 1990. 10

[GIS18] Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round MPC: Information-theoretic and black-box. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 123–151. Springer, Heidelberg, November 2018. 2

[GKPS18] Chaya Ganesh, Yashvanth Kondi, Arpita Patra, and Pratik Sarkar. Efficient adaptively secure zero-knowledge from garbled circuits. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 499–529. Springer, Heidelberg, March 2018. 4

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. ZKBoo: Faster zero-knowledge for Boolean circuits. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016*, pages 1069–1083. USENIX Association, August 2016. 3, 16

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 1

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. 1

[HM04] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 58–76. Springer, Heidelberg, February 2004. 1, 4

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007. 3, 12, 16

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of Structure in Complexity Theory. Tenth Annual IEEE Conference*, pages 134–147. IEEE, 1995. 4

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018. 3, 16, 17, 29, 30

[KZ20] Benjamin Kuykendall and Mark Zhandry. Towards non-interactive witness hiding. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 627–656. Springer, Heidelberg, November 2020. 10

[LR22] Anna Lysyanskaya and Leah Namisa Rosenbloom. Universally composable sigma-protocols in the global random-oracle model. *Cryptology ePrint Archive*, 2022. 4

[MR19] Daniel Masny and Peter Rindal. Endemic oblivious transfer. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 309–326. ACM Press, November 2019. 2

[MRS17] Payman Mohassel, Mike Rosulek, and Alessandra Scafuro. Sublinear zero-knowledge arguments for RAM programs. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 501–531. Springer, Heidelberg, April / May 2017. 1, 2, 3, 4

[MY04] Philip D. MacKenzie and Ke Yang. On simulation-sound trapdoor commitments. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 382–400. Springer, Heidelberg, May 2004. 3, 19

[Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Heidelberg, August 2003. 3, 4, 20, 31

[Rab05] Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. https://eprint.iacr.org/2005/187. 12

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982. 1

A Proof of Theorem 2

We first recall the theorem before giving the formal proof.

Theorem 2. *Assume the NIWH protocol* Π_{NIWH} *has a straight-line extractor* $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ *with knowledge error* $\varepsilon(\lambda)$ *and is witness hiding with adversarial advantage* $\mathbf{Adv}^{\mathsf{WH}}(\mathcal{A},\lambda)$*. Assume the equivocal commitment* ECom *is biding with adversarial advantage* $\mathbf{Adv}^{\mathsf{BIND}}(\mathcal{A},\lambda)$ *and is hiding with adversarial advantage* $\mathbf{Adv}^{\mathsf{HIDE}}(\mathcal{A},\lambda)$*. Assume* $\mathcal{G}_{\mathsf{oRO}}$ *is parameterized by the output length* $\ell_{\mathsf{out}}(\lambda)$*. Then the protocol* Π_{tCOM} *GUC-realizes functionality* $\mathcal{F}_{\mathsf{tCOM}}$ *in the* $\mathcal{G}_{\mathsf{oRO}}$ *hybrid world against static malicious corruption with advantage*

$$\max\{\varepsilon(\lambda)+(2-2\varepsilon(\lambda))\cdot\mathbf{Adv}^{\mathsf{EQUIV}}(\mathcal{A},\lambda),\mathbf{Adv}^{\mathsf{WH}}(\mathcal{A},\lambda)+\mathbf{Adv}^{\mathsf{BIND}}(\mathcal{A},\lambda)+2^{-\ell_{\mathsf{out}}(\lambda)}\}\ .$$

Proof. We now prove the security of our protocol Π_{tCOM} by showing it is a GUC-secure realization of $\mathcal{F}_{\mathsf{tCOM}}$. We only need to prove that Π_{tCOM} EUC-realizes $\mathcal{F}_{\mathsf{tCOM}}$ with respect to the shared functionality $\mathcal{G}_{\mathsf{oRO}}$. Therefore, we describe the workflow of $\mathcal{S}$ in the ideal-world with $\mathcal{F}_{\mathsf{tCOM}}$, and give a proof that the simulation in the ideal-world setting $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\mathcal{F}_{\mathsf{tCOM}},\mathcal{S},\mathcal{Z}}$ is computationally indistinguishable from a real-world execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\Pi_{\mathsf{tCOM}},\mathcal{A},\mathcal{Z}}$ for any PPT adversary $\mathcal{A}$ and any PPT $\mathcal{G}_{\mathsf{oRO}}$-constrained environment $\mathcal{Z}$.

Simulating Communication with $\mathcal{Z}$. The simulator $\mathcal{S}$ simply forwards the communication between $\mathcal{A}$ and $\mathcal{Z}$.

Simulating Two Honest Parties. Since we are in the secure channels model, $\mathcal{S}$ simply notifies $\mathcal{A}$ that communications (with messages of appropriate length) have taken place between the committer C and the receiver R.

Simulating the Honest Committer Against the Malicious Receiver. In this case, we suppose that the committer C is honest while the receiver R^* is statically corrupted. Here $\mathcal{S}$ needs to send a simulated commitment without knowing the message m in the committing phase, and open it to the message m in the opening phase. We describe the strategy of $\mathcal{S}$ as follows:

- Commitment Phase: Upon receiving $(\textsc{Receipt},\mathsf{sid},C,R)$ from $\mathcal{F}_{\mathsf{tCOM}}$, do
  - Round 1: Wait until the message (ck,π) arrives. Check if $\Pi_{\mathsf{NIWH}}.V(\mathsf{ck},\pi)=1$ holds. If not, abort; otherwise, extract $\mathsf{td}\leftarrow\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}(\mathsf{ck},\pi)$ such that $\mathsf{ECom.KeyVer}(\mathsf{ck},td)=1$. Abort if $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ fails.
  - Round 2: Generate equivocal commitments $(c_1,\mathsf{st}_1)\leftarrow\mathsf{ECom.EquCom}(\mathsf{ck},\mathsf{td})$, $(c_2,\mathsf{st}_2)\leftarrow\mathsf{ECom.EquCom}(\mathsf{ck},\mathsf{td})$. Send (c_1,c_2) to R^*.
- Opening Phase: Upon receiving $(\textsc{Decommit},\mathsf{sid},C,R,m)$ from $\mathcal{F}_{\mathsf{tCOM}}$, do
  - Round 3: Obtain $d_1\leftarrow\mathsf{ECom.Equiv}(\mathsf{ck},\mathsf{td},c_1,\mathsf{st}_1,m)$. Select a λ-bit random r and compute $h\leftarrow\mathsf{oRO}(\mathsf{sid},\text{`}C\text{'}||m||d_1||r)$. Obtain $d_2\leftarrow\mathsf{ECom.Equiv}(\mathsf{ck},\mathsf{td},c_2,\mathsf{st}_2,h)$. Send (m,d_1,d_2,r) to R^*.

We prove the indistinguishability through the following hybrid experiments:

- $\mathcal{H}_0$: This is the real-world execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\Pi_{\mathsf{tCOM}},\mathcal{A},\mathcal{Z}}$.
- $\mathcal{H}_1$: Same as $\mathcal{H}_0$, except that $\mathcal{S}$ aborts when $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ fails to extract a valid equivocation trapdoor td such that $\mathsf{ECom.KeyVer}(\mathsf{ck},td)=1$.

  Lemma 1. *If* Π_{NIWH} *is a NIWH argument that has a straight-line extractor* $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ *with knowledge error* $\varepsilon(\lambda)$*, then* $\mathcal{H}_1$ *is indistinguishable from* $\mathcal{H}_0$ *with adversarial advantage* $\varepsilon(\lambda)$*.*

  Proof. The probability of $\mathcal{S}$ aborting is equal to the knowledge error $\varepsilon(\lambda)$. Therefore, $\mathcal{H}_1$ is indistinguishable from $\mathcal{H}_0$ with adversarial advantage $\varepsilon(\lambda)$. □

- $\mathcal{H}_2$: Same as $\mathcal{H}_1$, except that $\mathcal{S}$ successfully uses $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ to find the valid td, generates the equivocal commitments (c_1, c_2) on dummy strings and equivocate (c_1, c_2) to new openings (d_1, d_2) with respect to (m, h), where $h = \mathsf{oRO}(\mathsf{sid}, `C'||m||d_1||r)$.

  Lemma 2. *If* Π_{NIWH} *is a NIWH argument that has a straight-line extractor* $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ *with knowledge error* $\varepsilon(\lambda)$*, and* ECom *is an equivocal commitment satisfying equivocal property with adversarial advantage* $\mathbf{Adv}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda)$*, then* $\mathcal{H}_2$ *is indistinguishable from* $\mathcal{H}_1$ *with adversarial advantage* $(2 - 2\varepsilon(\lambda)) \cdot \mathbf{Adv}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda)$*.*

  Proof. The probability of $\mathcal{S}$ extracting the valid td such that $\mathsf{ECom.KeyVer}(\mathsf{ck}, \mathsf{td}) = 1$ is $1 - \varepsilon(\lambda)$. Since the output of oRO is truly random, h sent by $\mathcal{S}$ is perfect indistinguishable from the one sent by the honest committer. The only thing left is to show the (c_1, c_2, d_1, d_2) are indistinguishable. In this case, we observe that if there is a PPT adversary $\mathcal{A}$ such that $\mathcal{Z}$ can distinguish $\mathcal{H}_1$ from $\mathcal{H}_0$, then we can construct a PPT $\mathcal{B}$ which breaks the equivocal property of the underlying ECom scheme. We first focus on the case concerning (c_1, d_1). During the reduction, $\mathcal{B}$ simulates $\mathcal{G}_{\mathsf{oRO}}$ and starts Π_{tCOM} with $\mathcal{A}$ by running $\mathcal{A}$ internally as black-box. First $\mathcal{B}$ waits for $\mathcal{A}$ to send the first round message (ck, π). Then $\mathcal{B}$ invokes $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ to extract the valid td. After that, $\mathcal{B}$ sends $(\mathsf{ck}, \mathsf{td})$ along with the message m to $\mathcal{C}$. After receiving (c_1, d_1) from $\mathcal{C}$, $\mathcal{B}$ simulates other messages and continues the protocol with $\mathcal{A}$. When $\mathcal{A}$ outputs a bit b, where $b = 0$ indicates (c_1, d_1) is the real commitment-opening pair and $b = 1$ indicates (c_1, d_1) is the equivocated one, $\mathcal{B}$ forwards the same bit b to $\mathcal{C}$. Clearly, $\mathcal{B}$ wins whenever $\mathcal{A}$ wins. The situation concerning (c_2, d_2) is similar. Therefore, we conclude that $\mathcal{H}_3$ is indistinguishable from $\mathcal{H}_2$ with adversarial advantage $(2 - 2\varepsilon(\lambda)) \cdot \mathbf{Adv}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda)$. □

Hybrid $\mathcal{H}_3$ is identical to the ideal world execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\mathcal{F}_{\mathsf{tCOM}}, \mathcal{S}, \mathcal{Z}}$. To conclude, when receiver is corrupted, the overall distinguishing advantage is at most $\varepsilon(\lambda) + (2 - 2\varepsilon(\lambda)) \cdot \mathbf{Adv}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda)$.

Simulating the Honest Receiver Against the Malicious Committer. In this case, we suppose that the committer C^* is statically corrupted while the receiver R is honest. Here $\mathcal{S}$ needs to extract the committed value from the commitment sent by C^* in the committing phase. We describe the strategy of $\mathcal{S}$ as follows:

- Committing Phase:
  - Round 1: Act as an honest receiver in Round 1 described in Π_{tCOM}.
  - Round 2: Upon receiving (c_1, c_2) from C^*, request the illegitimate queries from $\mathcal{F}_{\mathsf{tCOM}}$. After receiving $\mathcal{Q}_{\mathsf{sid}}$, check if there exists a query of the form $(\mathsf{sid}, `C'||m||d_1||r)$ such that $\mathsf{ECom.ComVer}(\mathsf{ck}, c_1, d_1) = 1$. If so, set $m' := m$; otherwise, set $m' := 0^\lambda$. Send $(\textsc{Commit}, \mathsf{sid}, C, R, m')$ to $\mathcal{F}_{\mathsf{tCOM}}$ on behalf the dummy committer.
- Opening Phase:
  - Round 3: Act as a honest receiver described in Round 3 in Π_{tCOM}. If the receiver algorithm accepts m^* from C^*, then check if $m^* = m'$. If so, send $(\textsc{Decommit}, \mathsf{sid}, C, R)$ to $\mathcal{F}_{\mathsf{tCOM}}$ on behalf the dummy committer; otherwise, abort.

We prove the indistinguishability through the following hybrid experiments:

- $\mathcal{H}_0$: This is the real-world execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\Pi_{\mathsf{tCOM}}, \mathcal{A}, \mathcal{Z}}$.
- $\mathcal{H}_1$: Same as $\mathcal{H}_0$, except that in the committing phase, $\mathcal{S}$ obtains $\mathcal{Q}_{\mathsf{sid}}$, determines m' and sends $(\textsc{Commit}, \mathsf{sid}, C, R, m')$ to $\mathcal{F}_{\mathsf{tCOM}}$. The perfect indistinguishability between $\mathcal{H}_0$ and $\mathcal{H}_1$ is trivial since $\mathcal{S}$ does not modify anything.
- $\mathcal{H}_2$: Same as $\mathcal{H}_1$, except that $\mathcal{S}$ aborts in the opening phase when $m^* \neq m'$.

  Lemma 3. *If* Π_{NIWH} *is a NIWH argument satisfying witness hiding property with adversarial advantage* $\mathbf{Adv}^{\mathsf{WH}}(\mathcal{A}, \lambda)$*,* ECom *is an equivocal commitment scheme satisfying binding property with adversarial advantage* $\mathbf{Adv}^{\mathsf{BIND}}(\mathcal{A}, \lambda)$*, and* $\mathcal{G}_{\mathsf{oRO}}$ *is parameterized by the output length* $\ell_{\mathsf{out}}(\lambda)$*, then* $\mathcal{H}_2$ *is indistinguishable from* $\mathcal{H}_1$ *with advantage* $\mathbf{Adv}^{\mathsf{WH}}(\mathcal{A}, \lambda) + \mathbf{Adv}^{\mathsf{BIND}}(\mathcal{A}, \lambda) + 2^{-\ell_{\mathsf{out}}(\lambda)}$*.*

Proof. Here $\mathcal{S}$ aborts when $m^* \neq m'$. This would happen when (i) C^* does not query $\mathcal{G}_{\mathsf{oRO}}$ on input $(\mathsf{sid}, `C'||m||d_1||r)$, but guesses the output of $\mathcal{G}_{\mathsf{oRO}}$ precisely; (ii) C^* is able to open the commitment sent earlier to any value. It is easy to see that the former case would happen at at a negligible probability, namely $2^{-\ell_{\mathsf{out}}(\lambda)}$.

Now let us focus on the latter case. The witness hiding property guarantees that the corrupted committer cannot obtain the td. Therefore, we observe that if there is a PPT adversary $\mathcal{A}$ such that $\mathcal{Z}$ can distinguish $\mathcal{H}_2$ from $\mathcal{H}_1$, then we can construct a PPT $\mathcal{B}$ which breaks the binding property of the underlying ECom. During the reduction, $\mathcal{B}$ simulates $\mathcal{G}_{\mathsf{oRO}}$ and starts Π_{tCOM} with $\mathcal{A}$ by running $\mathcal{A}$ internally as black-box. First $\mathcal{B}$ interacts with the binding game challenger $\mathcal{C}$, and receives ck from $\mathcal{C}$. Then $\mathcal{B}$ simulates the NIWH proof π and sends (ck, π) to $\mathcal{A}$. When $\mathcal{A}$ makes $\mathcal{S}$ aborts, i.e., $\mathcal{A}$ queries $\mathcal{G}_{\mathsf{oRO}}$ on input $m||d_1||r$, but later sends another valid opening d_1', $\mathcal{B}$ sends c_1, d_1, d_1' to $\mathcal{C}$. Clearly, $\mathcal{B}$ wins whenever $\mathcal{A}$ wins. We conclude that $\mathcal{H}_2$ is indistinguishable from $\mathcal{H}_1$ with adversarial advantage $\mathbf{Adv}^{\mathsf{WH}}(\mathcal{A}, \lambda) + \mathbf{Adv}^{\mathsf{BIND}}(\mathcal{A}, \lambda) + 2^{-\ell_{\mathsf{out}}(\lambda)}$. □

Hybrid $\mathcal{H}_3$ is identical to the ideal world execution $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{oRO}}}_{\mathcal{F}_{\mathsf{tCOM}}, \mathcal{S}, \mathcal{Z}}$. To conclude, when committer is corrupted, the overall distinguishing advantage is at most $\mathbf{Adv}^{\mathsf{WH}}(\mathcal{A}, \lambda) + \mathbf{Adv}^{\mathsf{BIND}}(\mathcal{A}, \lambda) + 2^{-\ell_{\mathsf{out}}(\lambda)}$.

In conclusion, the protocol Π_{tCOM} GUC-realizes the functionality $\mathcal{F}_{\mathsf{tCOM}}$ in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world against static malicious corruption with advantage $\max\{\varepsilon(\lambda) + (2 - 2\varepsilon(\lambda)) \cdot \mathbf{Adv}^{\mathsf{EQUIV}}(\mathcal{A}, \lambda), \mathbf{Adv}^{\mathsf{WH}}(\mathcal{A}, \lambda) + \mathbf{Adv}^{\mathsf{BIND}}(\mathcal{A}, \lambda) + 2^{-\ell_{\mathsf{out}}(\lambda)}\}$. □

B A Lower Bound on Round Complexity for GUC-Secure Commitment in the GPRO Model

Here we show that it is impossible to achieve GUC-secure commitment with one round for the committing phase in the $\mathcal{G}_{\mathsf{pRO}}$ hybrid world against static adversaries. We stress that this result is stronger than Theorem 1, since this result rules out the possibility of a commitment scheme which consists of one round for the committing phase and multiple rounds for the opening phase. We first provide a normal commitment functionality $\mathcal{F}_{\mathsf{COM}}$ in Figure 10, since $\mathcal{F}_{\mathsf{tCOM}}$ does not fit the $\mathcal{G}_{\mathsf{pRO}}$ hybrid world.

Functionality $\mathcal{F}_{\mathsf{COM}}$

The functionality interacts with two parties C, R and an adversary $\mathcal{S}$.

- Upon receiving $(\textsc{Commit}, \mathsf{sid}, C, R, m)$ from C, do:
  - Record the tuple (sid, C, R, m), and send $(\textsc{Receipt}, \mathsf{sid}, C, R)$ to R and $\mathcal{S}$.
  - Ignore any subsequent $\textsc{Commit}$ command.
- Upon receiving $(\textsc{Decommit}, \mathsf{sid}, C, R)$ from C, do:
  - If there is a tuple (sid, C, R, m) recorded, send $(\textsc{Decommit}, \mathsf{sid}, C, R, m)$ to R and $\mathcal{S}$, and halt.
  - Otherwise, ignore the message.

Figure 10: The Functionality $\mathcal{F}_{\mathsf{COM}}$ for Commitment

Theorem 6. *There exists no terminating single-message (only restricted in the committing phase) protocol Π that GUC-realizes $\mathcal{F}_{\mathsf{COM}}$ depicted in Figure 10 with static security, using only the shared functionality for global programmable random oracle $\mathcal{G}_{\mathsf{pRO}}$.*

Proof. We proceed by contradiction. Suppose there exists such a protocol Π that GUC-realizes $\mathcal{F}_{\mathsf{COM}}$ in the $\mathcal{G}_{\mathsf{pRO}}$ hybrid world. Then there must exist a PPT simulator $\mathcal{S}$ such that $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{pRO}}}_{\mathcal{F}_{\mathsf{COM}}, \mathcal{S}, \mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}^{\mathcal{G}_{\mathsf{pRO}}}_{\Pi, \mathcal{A}, \mathcal{Z}}$ for any PPT adversary $\mathcal{A}$ and any PPT $\mathcal{G}_{\mathsf{pRO}}$-externally constraint environment $\mathcal{Z}$.

In particular, let us consider the session with SID sid_1, and let $\mathcal{A}$ be a dummy adversary that simply forwards protocol flows between corrupt parties and the environment $\mathcal{Z}$. Let $\mathcal{Z}$ corrupt the committer C^* at first. Then $\mathcal{Z}$ chooses a random bit $b \in \{0, 1\}$ and instructs C to execute the honest committer algorithm to produce the commitment ψ on input b, which we denote by $\psi \leftarrow C(\mathsf{sid}_1, b, \mathcal{Q}_{\mathsf{sid}_1,C})$, where $\mathcal{Q}_{\mathsf{sid}_1,C}$ is the queries to $\mathcal{G}_{\mathsf{pRO}}$ sent by C^*. In the opening phase, $\mathcal{Z}$ instructs C^* to reveal the committed value b and waits for R to output b'. If $b = b'$, $\mathcal{Z}$ outputs 1; otherwise, $\mathcal{Z}$ outputs 0.

In order to make the GUC experiments above remain indistinguishable, the simulator $\mathcal{S}$ needs to extract the committed value b from the commitment ψ, and send $(\mathsf{Commit}, \mathsf{sid}_1, C, R, b)$ to $\mathcal{F}_{\mathsf{COM}}$ on behalf of the dummy committer. Recall that, the main advantage of $\mathcal{S}$ is that it can program the $\mathcal{G}_{\mathsf{pRO}}$ on unqueried points. If we denote by $\mathsf{Prog}_{\mathsf{sid}_1}$ the queries programmed by $\mathcal{S}$ and denote by $\mathcal{Q}_{\mathsf{sid}_1,\mathcal{S}}$ the queries sent by $\mathcal{S}$, then we can write $b \leftarrow \mathsf{ExtBit}(\mathsf{sid}_1, \psi, \mathsf{Prog}_{\mathsf{sid}_1}, \mathcal{Q}_{\mathsf{sid}_1,\mathcal{S}})$ to denote the event where $\mathcal{S}$ extracts the committed value b from ψ using $\mathsf{Prog}_{\mathsf{sid}_1}$ and $\mathcal{Q}_{\mathsf{sid}_1,\mathcal{S}}$. We note that, $\mathcal{S}$ should be able to handle any PPT adversary $\mathcal{A}$ and any PPT $\mathcal{Z}$. Consider such a case where $\mathcal{Z}$ queries $\mathcal{G}_{\mathsf{pRO}}$ everything that will be needed in advance, then starts the protocol Π and instructs C^* to run the honest committer algorithm on those previously queried points. In this case, we have $\Pr[\mathsf{Prog}_{\mathsf{sid}_1} \cap \mathcal{Q}_{\mathsf{sid}_1,C} \neq \emptyset] = 0$, and we denote by $b \leftarrow \mathsf{ExtBit}(\mathsf{sid}_1, \psi, \emptyset, \mathcal{Q}_{\mathsf{sid}_1,\mathcal{S}})$ the event where $\mathcal{S}$ is still able to extract b from ψ. We can regard ExtBit as a PPT algorithm. If we switch to the session with a different SID, this algorithm still works as long as we provide the appropriate input.

In the following, we show that the existence of the simulator $\mathcal{S}$ above contradicts the security of Π against static corruptions, by creating a particular environment $\mathcal{Z}'$ which succeeds in distinguishing $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{pRO}}}_{\mathcal{F}_{\mathsf{COM}},\mathcal{S}',\mathcal{Z}'}$ from $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{pRO}}}_{\Pi,\mathcal{A},\mathcal{Z}'}$ after a static corruption operation for any PPT simulator $\mathcal{S}'$. Now consider the session with SID sid_2. We let $\mathcal{Z}'$ corrupt the receiver R^* at first, and feed the honest committer C with a randomly selected bit $b \in \{0,1\}$, finally wait for R^* to output $(\textsc{Receipt}, \mathsf{sid}_2, C, R)$. In this case, the simulator $\mathcal{S}'$ needs to produces an equivocal commitment ψ without knowing b. If we denote by $\mathcal{Q}_{\mathsf{sid}_2,\mathcal{S}'}$ the queries sent by $\mathcal{S}'$ and $\mathsf{Prog}_{\mathsf{sid}_2}$ the queries programmed by $\mathcal{S}'$, then we can write $\psi \leftarrow \mathsf{FakeCom}(\mathsf{sid}_2, \mathcal{Q}_{\mathsf{sid}_2,\mathcal{S}'}, \mathsf{Prog}_{\mathsf{sid}_2})$ to denote the event where $\mathcal{S}'$ produces the equivocal commitment ψ using $\mathcal{Q}_{\mathsf{sid}_2,\mathcal{S}'}$ and $\mathsf{Prog}_{\mathsf{sid}_2}$. We note that, the entire computation of ψ is totally independent of b. After receiving ψ, $\mathcal{Z}'$ simply runs $b' \leftarrow \mathsf{ExtBit}(\mathsf{sid}_2, \psi, \emptyset, \mathcal{Q}_{\mathsf{sid}_2,\mathcal{S}})$. In the real-world, we always have $b' = b$. In the ideal-world, since the entire computation of ψ is independently of b, we have $b' = b$ with probability at most $\frac{1}{2}$. Therefore, $\mathcal{Z}'$ can distinguish between the real-world and the ideal-world experiments at least $\frac{1}{2}$, contradicting our assumption that Π is GUC-secure. □

C A Lower Bound on Round Complexity for GUC-Secure ZK in the GPRO Model

We show it is impossible to achieve GUC-secure NIZK protocols for *non-trivial* NP relations in the $\mathcal{G}_{\mathsf{pRO}}$ hybrid world against static adversaries. We first provide the ZK functionality $\mathcal{F}_{\mathsf{ZK}}$ in Figure 11.

Functionality $\mathcal{F}_{\mathsf{ZK}}$

The functionality interacts with two parties P, V and an adversary $\mathcal{S}$. It is parameterized by an NP relation $\mathcal{R}$.

- Upon receiving $(\text{ZK-}\textsc{prover}, \mathsf{sid}, P, V, x, w)$ from P, do:
  - Assert $\mathcal{R}(x, w) = 1$.
  - Output $(\text{ZK-}\textsc{proof}, \mathsf{sid}, P, V, x)$ to V and the adversary $\mathcal{S}$.

Figure 11: The Functionality $\mathcal{F}_{\mathsf{ZK}}$ for Zero-Knowledge

Definition 21. *We say an NP relation $\mathcal{R}$ defining the language $\mathcal{L}$ is non-trivial if there is no PPT algorithm efficiently decides membership in $\mathcal{L}$ (i.e. $\mathcal{L} \notin BPP$). Furthermore, we say $\mathcal{R}$ is non-trivial with respect to shared functionality $\mathcal{G}$ if there is no PPT algorithm efficiently decides membership in $\mathcal{L}$ even when given oracle access to $\mathcal{G}$.*

Theorem 7. *There exists no terminating one-round protocol Π that GUC-realizes $\mathcal{F}_{\mathsf{ZK}}$ depicted in Figure 11 with static security, using only the shared functionality for global programmable random oracle $\mathcal{G}_{\mathsf{pRO}}$, for any NP relation $\mathcal{R}$ that is non-trivial with respect to $\mathcal{G}_{\mathsf{pRO}}$.*

Proof. We proceed by contradiction. Suppose there exists such a protocol Π that GUC-realizes $\mathcal{F}_{\mathsf{ZK}}$ in the $\mathcal{G}_{\mathsf{pRO}}$ hybrid world for an NP relation $\mathcal{R}$ defining the language $\mathcal{L}$. Then there must exist a PPT simulator $\mathcal{S}$ such that $\mathsf{EXEC}^{\mathcal{G}_{\mathsf{pRO}}}_{\mathcal{F}_{\mathsf{ZK}},\mathcal{S},\mathcal{Z}} \overset{c}{\approx} \mathsf{EXEC}^{\mathcal{G}_{\mathsf{pRO}}}_{\Pi,\mathcal{A},\mathcal{Z}}$ for any PPT adversary $\mathcal{A}$ and any PPT $\mathcal{G}_{\mathsf{pRO}}$-externally constraint environment $\mathcal{Z}$.

Let us consider the session with SID sid_1, and let $\mathcal{A}$ be a dummy adversary that simply forwards protocol flows between corrupt parties and the environment $\mathcal{Z}$. Let $\mathcal{Z}$ corrupt the prover P^* at first. Then $\mathcal{Z}$ chooses $(x, w) \in \mathcal{R}$ as the input, instructs P to run the honest prover algorithm $\pi \leftarrow P(\mathsf{sid}_1, x, w, \mathcal{Q}_{\mathsf{sid}_1,P})$, where $\mathcal{Q}_{\mathsf{sid}_1,P}$ is the queries sent by P^*. Then $\mathcal{Z}$ waits for V to output x'. If $x = x'$, $\mathcal{Z}$ outputs 1; otherwise, $\mathcal{Z}$ outputs 0.

In order to make the GUC experiments above remain indistinguishable, the simulator $\mathcal{S}$ needs to extract a valid witness w such that $(x, w) \in \mathcal{R}$, and then sends $(\text{ZK-PROVER}, \mathsf{sid}, P, V, x, w)$ to $\mathcal{F}_{\mathsf{ZK}}$ on behalf of the dummy prover; otherwise, $\mathcal{Z}'$ can successfully distinguish the two experiments, contradicting the GUC-security of the protocol Π. We write $w \leftarrow \mathsf{ExtWit}(\mathsf{sid}_1, x, \pi, \mathsf{Prog}_{\mathsf{sid}_1}, \mathcal{Q}_{\mathsf{sid}_1,\mathcal{S}})$ to denote the event where $\mathcal{S}$ extracts the witness w from the proof π, where $\mathsf{Prog}_{\mathsf{sid}_1}$ is the queries programmed by $\mathcal{S}$ and $\mathcal{Q}_{\mathsf{sid}_1,\mathcal{S}}$ is the queries sent by $\mathcal{S}$. Analogous to the discussion of Theorem 6, $\mathcal{S}$ should handle the case where $\Pr[\mathsf{Prog}_{\mathsf{sid}_1} \cap \mathcal{Q}_{\mathsf{sid}_1,P} \neq \emptyset] = 0$, and we write $b \leftarrow \mathsf{ExtWit}(\mathsf{sid}_1, x, \pi, \emptyset, \mathcal{Q}_{\mathsf{sid}_1,\mathcal{S}})$ to denote the event where $\mathcal{S}$ is still able to extract b from π. Note that, the algorithm ExtWit can be run when we switch to another SID.

In the following, we consider another PPT $\mathcal{G}_{\mathsf{oRO}}$-externally constrained environment $\mathcal{Z}'$ and PPT simulator $\mathcal{S}'$. Now consider a session with SID sid_2. We let $\mathcal{Z}'$ corrupt the verifier V^* at first, and feed the honest prover P with $(x, w) \in \mathcal{R}$. Then $\mathcal{Z}$ waits for P to send the proof π. If π is valid, $\mathcal{Z}$ outputs 1; otherwise, $\mathcal{Z}$ outputs 0.

In this case, $\mathcal{S}'$ needs to produce an accepting proof π without w, since w is the hidden input to the honest party. More precisely, if we denote by $\tilde{\mathcal{Q}}_{\mathsf{sid}_2,\mathcal{S}'}$ the queries sent by $\mathcal{S}'$ and denote by $\mathsf{Prog}_{\mathsf{sid}_2}$ the queries programmed by $\mathcal{S}'$, then we can write $\pi \leftarrow \mathsf{FakeProof}(\mathsf{sid}_2, x, \tilde{\mathcal{Q}}_{\mathsf{sid}_2,\mathcal{S}'}, \mathsf{Prog}_{\mathsf{sid}_2})$ to denote the event where the simulator $\mathcal{S}'$ can produce the proof π without the witness w. We note that, there is nothing related to w in both $\tilde{\mathcal{Q}}_{\mathsf{sid}_2,\mathcal{S}'}$ and $\mathsf{Prog}_{\mathsf{sid}_2}$, and $\mathsf{FakeProof}$ can be run when we switch to the session with another SID.

To conclude the proof, we show there exists a PPT decider $\mathcal{D}$ that can efficiently compute the witness w for any given statement $x \in \mathcal{L}$ using only $\mathcal{G}_{\mathsf{pRO}}$. Given any statement x, $\mathcal{D}$ first select a party to act as P simulated by $\mathcal{S}'$ and a party to act as V^* controlled by $\mathcal{A}/\mathcal{Z}$. Then we let $\mathcal{S}'$ program $\mathcal{G}_{\mathsf{pRO}}$ on queries $\mathsf{Prog}_{\mathsf{sid}}$, and let $P/\mathcal{S}'$ run $\pi \leftarrow \mathsf{FakeProof}(\mathsf{sid}, x, \tilde{\mathcal{Q}}_{\mathsf{sid},\mathcal{S}'}, \mathsf{Prog}_{\mathsf{sid}})$ to produce the fake proof π. Finally, we let V^* run $\leftarrow \mathsf{ExtWit}(\mathsf{sid}, x, \pi, \emptyset, \mathcal{Q}_{\mathsf{sid},\mathcal{S}})$ to output the witness w. If ExtWit succeeds to output w such that $(x, w) \in \mathcal{R}$, $\mathcal{D}$ outputs 1 indicating $x \in \mathcal{L}$; otherwise, $\mathcal{D}$ outputs 0 indicating $x \notin \mathcal{L}$. It is easy to see that when $x \in \mathcal{L}$, $\mathcal{D}$ always outputs 1; when $x \notin \mathcal{L}$, $\mathcal{D}$ outputs 1 with only negligible probability. Therefore, we have successfully constructed a PPT decider $\mathcal{D}$ that efficiently decides membership in $\mathcal{L}$ (i.e. $\mathcal{L} \in BPP$) when given oracle access to $\mathcal{G}_{\mathsf{pRO}}$. This contradicts the non-triviality of $\mathcal{R}$ w.r.t. $\mathcal{G}_{\mathsf{pRO}}$. □

D Proof of Proposition 1

We first present the protocol from [KKW18] in Figure 12. We then recall and prove the Proposition 1.

Proposition 1. *Assume the n-party MPC protocol Π_{MPC} is $(n-1)$-private in the preprocessing model. Let m be the number of executions of preprocessing phase, where $m \geq 2$. The 3-round SHVZK protocol $\Pi^{\mathsf{KWW}}_{\mathsf{SHVZK}}$ depicted in Figure 12:*

- *cannot achieve k-special soundness, for $k \leq m$.*
- *satisfies k-special soundness, for $k \geq m+1$.*

Proof. In order to prove the proposition above, we have to show that (i) given any k distinct valid transcripts, where $k \geq m+1$, there exists a PPT algorithm that can efficiently recover the witness; (ii) there always exists a PPT adversary $\mathcal{A}$ that can produce k distinct valid transcripts without knowing the witness, where $k \leq m$.

We first discuss the former case. Recall that the challenge of the protocol is (c, p), and the domain of the first challenge c is $[m]$. Therefore, given k distinct transcripts where $k \geq m+1$, there must exist a $c^* \in [m]$ and $p \neq p'$ such that (c^*, p) and (c^*, p') are the challenges that corresponds to two of the

Protocol $\Pi_{\mathsf{SHVZK}}^{\mathsf{KWW}}$

Primitives: n-party MPC protocol Π_{MPC} which realizes $f_{\mathcal{R}}$ with $(n-1)$-privacy in the preprocessing model, where $f_{\mathcal{R}}(x, w_1, \cdots, w_n) = \mathcal{R}(x, w_1 \oplus \cdots \oplus w_n)$. Statically binding commitment COM.
Inputs: P, V both have a public input x and an NP relation $\mathcal{R}$. P has a private input w such that $\mathcal{R}(x, w) = 1$.

Protocol:

- $\mathcal{P}_1(x, w, r)$:
  - For $i \in [m]$:
    - Derive λ-bit random seed_i from randomness r and generate $\{\mathsf{state}_{i,j}\}_{j\in[n]} \leftarrow \mathsf{Preprocess}(\mathsf{seed}_i)$.
    - Simulate the execution of Π_{MPC} using (x, w) and the states $\{\mathsf{state}_{i,j}\}_{j\in[n]}$, and output the views of the parties $\{\mathsf{view}_{i,j}(x, w_j)\}_{j\in[n]}$.
    - For $j \in [n]$: select λ-bit randomness $r_{i,j}, \widetilde{r}_{i,j}$, and compute $\mathsf{com}_{i,j} \leftarrow \mathsf{COM}(\mathsf{state}_{i,j}, r_{i,j})$ and $\widetilde{\mathsf{com}}_{i,j} \leftarrow \mathsf{COM}(\mathsf{view}_{i,j}(x, w_j), \widetilde{r}_{i,j})$.
  - Send $a := (\{\mathsf{com}_{i,j}, \widetilde{\mathsf{com}}_{i,j}\}_{i\in[m], j\in[n]})$.
- $\mathcal{V}_1(1^\lambda)$: Send $e := (c, p)$, where c is uniformly random in $[m]$ and p is uniformly random in $[n]$.
- $\mathcal{P}_2(x, w, r, e)$: Send $z := (\{\mathsf{state}_{i,j}, r_{i,j}\}_{i\neq c, j\in[n]}, \{\mathsf{state}_{c,j}, \mathsf{view}_{c,j}(x, w_j), \tilde{r}_{c,j}\}_{j\neq p})$.
- $\mathcal{V}_2(x, a, e, z)$: Output 1 if and only if the following checks pass:
  - Check the commitments are opened correctly:
    - For $i \in [m]/c, j \in [n]$: check $\mathsf{com}_{i,j} = \mathsf{COM}(\mathsf{state}_{i,j}, r_{i,j})$ holds.
    - For $j \in [n]/p$: check $\mathsf{com}_{c,j} = \mathsf{COM}(\mathsf{state}_{c,j}, r_{c,j})$ and $\widetilde{\mathsf{com}}_{c,j} = \mathsf{COM}(\mathsf{view}_{c,j}(x, w_j), \tilde{r}_{i,j})$ hold.
  - Check the correctness of the executions of the preprocessing phase:
    - For $i \in [m]/c$: check $\{\mathsf{state}_{i,j}\}_{j\in[n]}$ are well-formed.
  - Check the consistency between the opened views:
    - For $j \in [n]/p$: check $\mathsf{view}_{c,j}(x, w_j)$ follows from the $\mathsf{state}_{c,j}$ correctly and $\mathsf{view}_{c,j}(x, w_j)$ yields output 1.
    - Check $\{\mathsf{view}_{c,j}(x, w_j)\}_{j\in[n]/p}$ are consistent with each other.

Simulation: Given the challenge $e := (c, p)$ ahead, we have

- $\mathsf{Sim}(x, e, r)$:
  - For $i \in [m]$:
    - Derive λ-bit random seed_i from randomness r and generate $\{\mathsf{state}_{i,j}\}_{j\in[n]} \leftarrow \mathsf{Preprocess}(\mathsf{seed}_i)$.
    - Run the simulator algorithm of Π_{MPC} using x, p and the states generated by the c-th preprocessing phase (i.e., $\{\mathsf{state}_{c,j}\}_{j\in[n]}$), and output the simulated views $\{\mathsf{view}_{c,j}\}_{j\in[n]/p}$. Sample a random $\mathsf{view}_{c,p}$ of appropriate length.
    - For $j \in [n]$: select λ-bit randomness $r_{i,j}, \widetilde{r}_{i,j}$, and compute $\mathsf{com}_{i,j} \leftarrow \mathsf{COM}(\mathsf{state}_{i,j}, r_{i,j})$ and $\widetilde{\mathsf{com}}_{i,j} \leftarrow \mathsf{COM}(\mathsf{view}_{i,j}(x, w_j), \widetilde{r}_{i,j})$.
  - Output $a := (\{\mathsf{com}_{i,j}, \widetilde{\mathsf{com}}_{i,j}\}_{i\in[m], j\in[n]})$ and $z := (\{\mathsf{state}_{i,j}, r_{i,j}\}_{i\neq c, j\in[n]}, \{\mathsf{state}_{c,j}, \mathsf{view}_{c,j}(x, w_j), \tilde{r}_{c,j}\}_{j\neq p})$.

Figure 12: Protocol $\Pi_{\mathsf{SHVZK}}^{\mathsf{KWW}}$ From [KKW18]

given transcripts. For transcript that corresponds to (c^*, p), it reveals $\{\mathsf{state}_{c^*,j}, \mathsf{view}_{c^*,j}(x, w_j)\}_{j \neq p}$; and for transcript that corresponds to (c^*, p'), it reveals $\{\mathsf{state}_{c^*,j}, \mathsf{view}_{c^*,j}(x, w_j)\}_{j \neq p'}$. Since $p \neq p'$, we obtain $\{\mathsf{state}_{c^*,j}, \mathsf{view}_{c^*,j}(x, w_j)\}_{j \in [n]}$. Then we yield the private input w_j from the $\mathsf{state}_{c^*,j}, \mathsf{view}_{c^*,j}(x, w_j)$, and finally computes $w \leftarrow w_1 \oplus \cdots \oplus w_n$. Therefore, there exists a PPT algorithm that can efficiently recover the witness w when given $m+1$ distinct valid transcripts.

We then discuss the latter case. For $k \leq m$, we can construct a PPT adversary $\mathcal{A}$ that produces k distinct accepting transcripts without knowing the witness w as follows:

- Sample uniformly random $p \in [n]$.
- For $i \in [k]$:
  - Sample λ-bit randomness seed_i and generate $\{\mathsf{state}_{i,j}\}_{j \in [n]} \leftarrow \mathsf{Preprocess}(\mathsf{seed}_i)$.
  - Run the simulator algorithm of Π_{MPC} using x, p and the states generated by the i-th preprocessing phase (i.e., $\{\mathsf{state}_{i,j}\}_{j \in [n]}$), and output the simulated views $\{\mathsf{view}_{i,j}\}_{j \in [n]/p}$. Sample a random $\mathsf{view}_{i,p}$ of appropriate length.
  - For $j \in [n]$: select λ-bit randomness $r_{i,j}, \widetilde{r}_{i,j}$, and compute $\mathsf{com}_{i,j} \leftarrow \mathsf{COM}(\mathsf{state}_{i,j}, r_{i,j})$ and $\widetilde{\mathsf{com}}_{i,j} \leftarrow \mathsf{COM}(\mathsf{view}_{i,j}(x, w_j), \widetilde{r}_{i,j})$.
  - Set $e_i := (i, p)$ and $z_i := (\{\mathsf{state}_{k,j}, r_{k,j}\}_{k \neq i, j \in [n]}, \{\mathsf{state}_{i,j}, \mathsf{view}_{i,j}(x, w_j), \tilde{r}_{i,j}\}_{j \neq p})$.
- Set $a := (\{\mathsf{com}_{i,j}, \widetilde{\mathsf{com}}_{i,j}\}_{i \in [m], j \in [n]})$, and output $\{(a, e_i, z_i)\}_{i \in [m]}$.

It is easy to see that (a, e_i, z_i) is an accepting transcript for every $i \in [k]$, and $\{e_i := (i, p)\}_{i \in [k]}$ are distinct with each other. Therefore, there exists a PPT adversary $\mathcal{A}$ that can produce k distinct valid transcripts without knowing w, where $k \leq m$. In other words, given only k distinct valid transcripts where where $k \leq m$, it is not guaranteed that we can recover the witness from them.

In conclusion, we prove that $\Pi^{\mathsf{KWW}}_{\mathsf{SHVZK}}$: (i) cannot achieve k-special soundness, for $k \leq m$; (ii) satisfies k-special soundness, for $k \geq m+1$. $\square$

E Straight-line Extractable NIWH Argument

In [Pas03], Pass showed how to transform a 2-special sound SHVZK protocol into a straight-line extractable NIWH argument in the observable RO model. Here we generalize the idea of Pass and construct the straight-line extractable NIWH argument from k-special sound SHVZK protocol where $k \geq 2$.

Let P be the prover algorithm and V be the verifier algorithm. Let $\ell = \min\{\log_k(2^{\ell_{\mathsf{out}}(\lambda)}), \log_k(2^\lambda)\}$ be the repetition parameter. Formally, we present our protocol Π_{NIWH} in Figure 13 and prove the security through Theorem 8. If we use our SHVZK-PoK protocol Π_{SHVZK} depicted in Figure 8 as the main building block, then we can obtain a NIWH argument via MiniCrypt in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world.

Theorem 8. *Assume* OWF *is a one-way function that is hard to invert. Assume* Π_{SHVZK} *is a* k*-special sound SHVZK protocol. The protocol* Π_{NIWH} *described in Figure 13 is a NIWH argument which is witness hiding and is straight-line extractable.*

Proof. Witness Hiding. We first prove our protocol is witness hiding. We proceed by contradiction. Assume there exists a PPT adversary $\mathcal{A}$ that breaks the witness hiding property of Π_{NIWH} with non-negligible probability, then we are able to build a reduction $\mathcal{B}$ which violates the Hard to Invert (HI) property of the underlying OWF. First $\mathcal{B}$ interacts with the HI game challenger $\mathcal{C}$, and receives y from $\mathcal{C}$. Then $\mathcal{B}$ simulates $\mathcal{G}_{\mathsf{oRO}}$ and starts the protocol Π_{NIWH} with $\mathcal{A}$ by running $\mathcal{A}$ internally as black-box. Thus, our $\mathcal{B}$ sees all queries $\mathcal{A}$ makes to $\mathcal{G}_{\mathsf{oRO}}$ and produces their answers. The internal description of $\mathcal{B}$ follows:

- Select a random $h \in \{0,1\}^{\ell_{\mathsf{out}}(\lambda)}$, convert h into a k-ary number, and use h_i to represent the i-th digit of this k-ary number.
- For $i \in [\ell]$: select λ-bit random r_i, e_{i,h_i+1}, and run $(a_i, z_{i,h_i+1}) \leftarrow \Pi_{\mathsf{SHVZK}}.\mathsf{Sim}(y, e_{i,h_i+1}, r_i)$ for relation $\{(y, \mathsf{seed}) \mid y = \mathsf{OWF}(\mathsf{seed})\}$. Select λ-bit random $\{e_{i,j}, z_{i,j}\}_{j \in [k], j \neq h_i+1}$ such that $\{e_{i,j}\}_{j \in [k]}$ are distinct.

Protocol Π_{NIWH}

Primitives: SHVZK protocol $\Pi_{\mathsf{SHVZK}} := (\mathcal{P}_1, \mathcal{P}_2, \mathcal{V}_1, \mathcal{V}_2)$ and one-way function OWF.
Inputs: P and V has a common y. P has a private seed such that $y = \mathsf{OWF}(\mathsf{seed})$.

Proof Generation: $P(y, \mathsf{seed})$:

- For $i \in [\ell]$:
  - Select λ-bit random $r_i, \{s_{i,j}\}_{j\in[k]}$, and compute $a_i \leftarrow \Pi_{\mathsf{SHVZK}}.\mathcal{P}_1(y, \mathsf{seed}, r_i)$ for relation $\mathcal{R}_1 := \{(y, \mathsf{seed}) \mid y = \mathsf{OWF}(\mathsf{seed})\}$.
  - Select k distinct λ-bit random challenge $\{e_{i,j}\}_{j\in[k]}$.
  - For $j \in [k]$: compute the corresponding response $z_{i,j} \leftarrow \Pi_{\mathsf{SHVZK}}.\mathcal{P}_2(y, \mathsf{seed}, r_i, e_{i,j})$ for relation $\mathcal{R}_1$, and commit to the response $\mathsf{com}_{i,j} \leftarrow \mathsf{oRO}(\mathsf{sid}, `P'||z_{i,j}||s_{i,j})$.
- Set $a := (\{a_i, \{e_{i,j}, \mathsf{com}_{i,j}\}_{j\in[k]}\}_{i\in[\ell]})$, and compute $h \leftarrow \mathsf{oRO}(\mathsf{sid}, `P'||y||a)$.
- Convert h into a k-ary number, and use h_i to represent the i-th digit of this k-ary number.
- For $i \in [\ell]$: set $z_i := z_{i,h_i+1}$ and $s_i := s_{i,h_i+1}$. Send $\pi := (a, z := (\{z_i, s_i\}_{i\in[\ell]}))$.

Verification: $V(y, \pi)$:

- Compute $h \leftarrow \mathsf{oRO}(\mathsf{sid}, `P'||y||a)$, convert h into a k-ary number, and use h_i to represent the i-th digit of this k-ary number.
- For $i \in [\ell]$:
  - Check if $\{e_{i,j}\}_{j\in[k]}$ are distinct.
  - Check if the following conditions hold: $\mathsf{com}_{i,h_i+1} = \mathsf{oRO}(\mathsf{sid}, `P'||z_i||s_i)$ and $\Pi_{\mathsf{SHVZK}}.\mathcal{V}_2(y, a_i, e_{i,h_i+1}, z_i) = 1$ for relation $\mathcal{R}_1$.
- If all the checks pass, output 1; otherwise, output 0.

Figure 13: Protocol Π_{NIWH} in the GORO Model for Proving $y = \mathsf{OWF}(\mathsf{seed})$

- For $i \in [\ell], j \in [k]$: select λ-bit random $s_{i,j}$, and compute $\mathsf{com}_{i,j} \leftarrow \mathsf{oRO}(\mathsf{sid}, `P'||z_{i,j}||s_{i,j})$.
- Set $a := (\{a_i, \{e_{i,j}, \mathsf{com}_{i,j}\}_{j\in[k]}\}_{i\in[\ell]})$, and program the answer of $\mathcal{G}_{\mathsf{oRO}}$ as h on query $(\mathsf{sid}, `P'||y||a)$. Set $z := (\{z_{i,h_i+1}, s_{i,h_i+1}\}_{i\in[\ell]})$.
- Send $(\mathsf{sid}, y, \pi := (a, z))$ to $\mathcal{A}$, and wait for $\mathcal{A}$ to output seed^*.

When $\mathcal{A}$ outputs seed^*, $\mathcal{B}$ checks if $y = \mathsf{OWF}(\mathsf{seed}^*)$ holds. If so, $\mathcal{B}$ sends seed^* to the HI game challenger $\mathcal{C}$ and wins the game. Therefore, we prove that our protocol is witness-hiding.

Straight-line Extractor. We first show the strategy of the straight-line extractor $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$, on input (y, π) and illegitimate queries $\mathcal{Q}_{\mathsf{sid}}$, as follows:

- For $i \in [\ell]$: check if there exists a query of the form $(\mathsf{sid}, `P'||z_{i,j}||s_{i,j})$ such that $\mathsf{com}_{i,j} = \mathsf{oRO}(\mathsf{sid}, `P'||z_{i,j}||s_{i,j})$ for $j \in [k]$. If not, start over with the next i. Otherwise, extract $\mathsf{seed}_i \leftarrow \Pi_{\mathsf{SHVZK}}.\mathsf{Ext}(y, \{(a_{i,j}, e_{i,j}, z_{i,j})\}_{j\in[k]})$. If $\Pi_{\mathsf{SHVZK}}.\mathsf{Ext}$ succeeds to extract seed_i such that $y = \mathsf{OWF}(\mathsf{seed}_i)$, output $\mathsf{seed} := \mathsf{seed}_i$; otherwise, start over with the next i.
- Abort if all the steps above fails.

We then show the knowledge error of $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ is negligible. The knowledge error is exactly the probability of $\Pi_{\mathsf{NIWH}}.\mathsf{SLExt}$ aborting. Note that the straight-line extractor aborts when there exists a PPT $\mathcal{A}$ (i) guesses h_i correctly, and runs $(a_i, z_{i,h_i+1}) \leftarrow \Pi_{\mathsf{SHVZK}}.\mathsf{Sim}(y, e_{i,h_i+1}, r_i)$ to simulate the accepting proof; (ii) can not guess h_i correctly, thus, has to honestly commit to the response $z_{i,j}$, but $\Pi_{\mathsf{SHVZK}}.\mathsf{Ext}$ fails to output the valid seed_i such that $y = \mathsf{OWF}(\mathsf{seed}_i)$ for all $i \in [\ell]$. Due to the unpredictability of the output of $\mathcal{G}_{\mathsf{oRO}}$, the former case happens at probability $k^{-\ell} = \max\{2^{-\ell_{\mathsf{out}}(\lambda)}, 2^{-\lambda}\}$ which is negligible. Now let us focus on the latter case. We observe that if such $\mathcal{A}$ exists, we are able to build a reduction $\mathcal{B}$ which, by using $\mathcal{A}$ as a black-box, violates the k-Special Sound (k-SS) property of the underlying Π_{SHVZK}. First $\mathcal{B}$ interacts with the k-SS game challenger $\mathcal{C}$. In order to win the game, $\mathcal{B}$ simulates $\mathcal{G}_{\mathsf{oRO}}$ and starts the protocol Π_{NIWH} with

$\mathcal{A}$ by running $\mathcal{A}$ internally as black-box. When $\mathcal{A}$ outputs an accepting proof π, $\mathcal{B}$ looks up the query-answer table of $\mathcal{G}_{\mathsf{oRO}}$. With overwhelming probability, $\mathcal{B}$ finds at least one set of transcripts $\{(a_i, e_{i,j}, z_{i,j})\}_{j\in[k]}$ such that $\forall j \in [k] : \Pi_{\mathsf{SHVZK}}.\mathcal{V}_2(y, a_i, e_{i,j}, z_{i,j}) = 1$ and $\Pi_{\mathsf{SHVZK}}.\mathsf{Ext}$ fails to extract the valid witness. Finally $\mathcal{B}$ sends $(y, \{(a_i, e_{i,j}, z_{i,j})\}_{j\in[k]})$ to the k-SS game challenger $\mathcal{C}$ and wins the game. Therefore, we prove that our protocol has a straight-line extractor with negligible knowledge error. $\square$