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

Zhelei Zhou[1,2,**], Bingsheng Zhang[1,2,*,**], Hong-Sheng Zhou[3,*,***], and Kui Ren[1,2]

[1] Zhejiang University, {zl_zhou,bingsheng,kuiren}@zju.edu.cn
[2] ZJU-Hangzhou Global Scientific and Technological Innovation Center
[3] Virginia Commonwealth University, hszhou@vcu.edu

Abstract. In the UC framework, protocols 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 using global random oracles (GRO) as the trusted setup. In particular, we show that it is impossible to have a 2-round (1-round committing and 1-round opening) 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. Furthermore, we also examine the complete picture on round complexity of the GUC-secure commitments in various global RO models.

1 Introduction

Secure multi-party computation (MPC) [39,26] is one of the most important cornerstone of modern cryptography. It enables 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 in the literature. Typically, the security properties of an MPC protocol are formalized using the well-known "simulation-paradigm" [27,26]. 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 players provide their inputs to a trusted third party who computes f for them and relays the result back to the players.

* Corresponding authors: Bingsheng Zhang and Hong-Sheng Zhou.

** Work supported by the National Key R&D Program of China (No. 2021YFB3101601), the National Natural Science Foundation of China (Grant No. 62072401), "Open Project Program of Key Laboratory of Blockchain and Cyberspace Governance of Zhejiang Province", and 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.

From UC to GUC. To facilitate modular protocol design and analysis in the complex network environments, Canetti proposed the Universal Composibility (UC) framework [10], where, 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 the security of the MPC protocols under *concurrent executions*, and even other *arbitrary* protocols running in the same network cannot be adversarially affected — roughly speaking, the environment represents the collection of any other concurrent protocols. Additionally, this notion is closed under composition, enabling modular analysis of protocols.

However, protocols in the UC framework must be *subroutine respecting*, and shared setup cannot be directly modeled by the basic UC notion. To address this drawback, Canetti, Dodis, Pass and Walfish proposed the Generalized Universal Composibility (GUC) framework in 2007 [11]. Since then, many interesting and efficient protocols have been designed and analyzed under the GUC framework [20,14,37,9,15].

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 [12,11] 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. Random Oracle (RO) is a classic idealized setup that can be used to design UC-secure [28] and GUC-secure multi-party computation protocols [14,9].

Random oracle model [4] is a popular idealized model that has been widely used to justify the security of efficient cryptographic protocols. In spite of its known inability to provide provable guarantees when RO is instantiated with a real-world hash function [13], 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 other distributed system setting. Many RO-based non-interactive ZK systems, e.g., zk-STARK [5] and Fractal [17], are developed and deployed in real application scenarios. Note that, those RO-based protocols can achieve *post-quantum* security.

A natural formulation of a global RO, denoted as $\mathcal{G}_{\mathsf{sRO}}$, has been defined in [11]: it is accessible to all parties both in the ideal world and the real world, but it offers neither "observability" nor "programmability". We emphasize that, it has been proven that it is impossible to achieve GUC-secure commitment in the $\mathcal{G}_{\mathsf{sRO}}$ model [11]. Later, Canetti, Jain, and Scafuro [14] proposed a strengthened version of the 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.* [9] 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 by the adversary,

and we denote this strengthened version of the global RO as $\mathcal{G}_{\mathsf{pRO}}$. On top of both $\mathcal{G}_{\mathsf{oRO}}$ and $\mathcal{G}_{\mathsf{pRO}}$, Camensich *et al.* [9] then introduced an even stronger variant, called $\mathcal{G}_{\mathsf{poRO}}$, and they constructed a round-optimal GUC-secure commitment in the $\mathcal{G}_{\mathsf{poRO}}$ model [9]. Figure 3 depicts the relation of these global RO models.

Problem statement. We study the round complexity of GUC-secure commitment in the global RO models. Clearly protocols relying on a *less idealized* setup and *weaker* computational assumptions will allow us to gain better confidence in the proved security statement. Note that, round-optimal GUC secure commitments can be constructed based on the strong global RO setup $\mathcal{G}_{\mathsf{poRO}}$ [9]. On the other hand, in [11], it has been proven that constructing a GUC-secure commitment in the $\mathcal{G}_{\mathsf{sRO}}$ model is impossible. Between these two extremes, in [14], Cannetti et al have shown that it is feasible to construct a GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model; however, their construction relies on the discrete logarithm assumption, which cannot achieve (post-) quantum security. We are interested in GUC-secure commitment protocols using a global RO setup and Minicrypt [29] assumptions; these protocols can additionally achieve post-quantum security. This leads us to a natural research question:

> *What is the lower bounds of the round complexity*[4] *of a GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model?*

If there exists such a lower bound on the round complexity of a GUC-secure commitment in the $\mathcal{G}_{\mathsf{oRO}}$ model, we would like to find a round-optimal construction. We hereby 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. Our findings can be summarized as follows.

A new impossibility result in the $\mathcal{G}_{\mathsf{oRO}}$ model. In this work, we show that 2-round (1-round for committing and 1-round for opening) 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

[4] 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 [24,36].

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 $\mathcal{G}_{\mathsf{oRO}}$ 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; Therefore, 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-round for committing and 1-round for opening) 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 with 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 1st round, the receiver picks the commitment key and sends it to the committer; and (ii) in the 2nd round, the committer uses the equivocal commitment scheme to commit the message. To deploy this approach, the following questions need to be resolved:

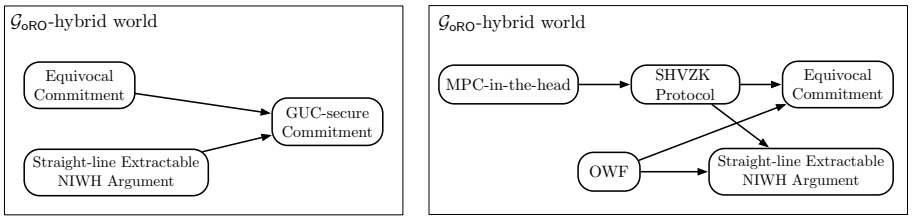
- *How to instantiate such a perfect-hiding non-interactive equivocal commitment?*
- *How can the simulator obtain the equivocation trapdoor?*

In [14] and [37], 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 discrete logarithm assumption which is not (post-) quantum secure. In this work, we show how to construct a candidate of the equivocal commitment only using Minicrypt assumptions, i.e. the existence of one-way functions, in the $\mathcal{G}_{\mathsf{oRO}}$ model.

To address the latter question, [14] introduced a 5-round mechanism that enables the simulator to obtain the equivocation trapdoor in the $\mathcal{G}_{\mathsf{oRO}}$ model; whereas, [37] proposed a more round-efficient (3-round) mechanism to do so. More precisely, [37] 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 1st round. Note that straight-line extractability is needed for this approach.

Following the technique proposed in [37], our framework adopts the Non-Interactive Witness Hiding (NIWH) argument with straight-line extractability [38] to prove the knowledge of equivocation trapdoor w.r.t. the commitment

key. The straight-line extractable NIWH argument 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.



(a) A general framework. (b) Instantiation of building blocks.

Fig. 1. Technique Roadmap

Non-interactive equivocal commitment in Minicrypt. As shown in [18,35], it is possible to 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 protocol, the prover sends the message flow a in the 1st round, and the receiver sends a public-coin randomness e as the challenge in the 2nd 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 $\mathcal{R}_{\mathcal{L}}$ be an NP relation whose associate language is $\mathcal{L}$. The receiver randomly samples a pair $(x, w) \in \mathcal{R}_{\mathcal{L}}$ 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 send m, z to the receiver, who will accept it if and only if (a, m, z) is an accepting SHVZK proof transcript. The equivocation trapdoor is w, which can be extracted from the straight-line extractor of NIWH as described above.

Since we aim to construct a commitment under Minicrypt assumptions, in our construction, $\mathcal{R}_{\mathcal{L}}$ is instantiated with a one-way function relation, i.e., $g(x) = y$ where g is a one-way function. Next, how to construct a 2-special sound SHVZK protocol under Minicrypt assumptions? One possible approach is to use the "MPC-in-the-head" paradigm proposed by Ishai *et al.* [30]. Roughly speaking, the main idea is for the prover to simulate the execution of an n-party computation protocol that checks if $(x, w) \in \mathcal{R}_{\mathcal{L}}$, 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 revealed views are consistent. Unfortunately, to the best of our knowledge, none of the followups [25,16,1,31,19] since the ini-

tial work of [30] can lead to a 2-special sound SHVZK protocol merely under 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).

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 under 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 [11], and round-optimal GUC-secure commitment can be constructed without further assumptions in the $\mathcal{G}_{\mathsf{poRO}}$ model [9]. What about the $\mathcal{G}_{\mathsf{pRO}}$? In this work, we also show some impossibility result: there exists *no* GUC-secure commitment with 1-round committing in the $\mathcal{G}_{\mathsf{pRO}}$ model (see details in the full version of our paper). However, the feasibility of round-optimal GUC-secure commitment under 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 [38]. 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 (see details in the full version of our paper).

1.2 Related Work

In terms of UC security with local setups, non-interactive commitments (1-round for committing and 1-round for opening) can be constructed under various setup assumptions. For instance, Canetti and Fischlin gave a candidate in the CRS model [12]; Hofheinz and Müller-Quade suggested a candidate in the RO model [28]. 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., [20] is dedicated to this research area. In [11], 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 Agumented CRS (ACRS) model. Later, Dodis *et al.* proved that there exists *no* GUC-secure commitment with 1-round committing phase in the ACRS model against adaptive adversaries [20]. 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 [2]. To bypass this impossibility result, $\mathcal{G}_{\mathsf{oRO}}$,$\mathcal{G}_{\mathsf{pRO}}$ and $\mathcal{G}_{\mathsf{poRO}}$ are proposed; note that 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 [14]. 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 1-round setup phase followed by a 2-round commitment [37]. Note that their construction also employed Pedersen commitment, which cannot achieve (post-) quantum security. Byali *et al.* gave a 2-round GUC-secure commitment construction in the CRS and $\mathcal{G}_{\mathsf{oRO}}$ hybrid model [8]. Following Byali *et al.* paradigm, GUC-secure ZK protocols [23,34] can also be constructed in the CRS and $\mathcal{G}_{\mathsf{oRO}}$ hybrid model. With regard to post-quantum security, [7] gave a 5-round lattice-based GUC-secure commitment and [6] 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 [9]. Recently, Canetti *et al.* proposed a 2-round OT adaptive-secure OT from DDH assumption in the $\mathcal{G}_{\mathsf{pRO}}$ model [15], 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 [3].

2 Preliminaries

2.1 Notations

Let $\lambda \in \mathbb{N}$ be the security parameter. We say that a function $\mathsf{negl} : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large λ, it holds that $\mathsf{negl}(\lambda) < \frac{1}{p(\lambda)}$. We write $y := \mathsf{Alg}(x; r)$ when the algorithm Alg on input x and randomness r, outputs y. We write $y \leftarrow \mathsf{Alg}(x)$ for the process of sampling the randomness r and setting $y := \mathsf{Alg}(x; r)$. We also write $y \leftarrow S$ for sampling y uniformly at random from the set S. 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 often write $\mathcal{R}_{\mathcal{L}}$ to denote the NP relation whose associate language is $\mathcal{L}$ for short. We also use $\mathcal{R}_{\mathcal{L}}(x, w) = 1$ to refer to $(x, w) \in \mathcal{R}_{\mathcal{L}}$. We say that two distribution ensembles $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ are identical (resp. computationally indistinguishable), denoted by $\mathcal{X} \equiv \mathcal{Y}$ (resp., $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$), if for any unbounded (resp., PPT) distinguisher $\mathcal{D}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|\Pr[\mathcal{D}(\mathcal{X}_\lambda) = 1] - \Pr[\mathcal{D}(\mathcal{Y}_\lambda) = 1]| = 0$ (resp., $\mathsf{negl}(\lambda)$). When we define a protocol/scheme in form of $\Pi = \Pi.\{\mathsf{Alg\text{-}1}, \ldots, \mathsf{Alg\text{-}n}\}$, we use the notation $\Pi.\mathsf{Alg\text{-}i}$ to refer to the algorithm $\mathsf{Alg\text{-}i}$ of Π where $\mathsf{Alg\text{-}i} \in \{\mathsf{Alg\text{-}1}, \ldots, \mathsf{Alg\text{-}n}\}$.

2.2 Universal Composability

Canetti's UC framework. The UC framework proposed by Canetti [10] 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, a protocol Π is defined to be a computer program (or several

programs) which is intended to be executed by 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.

Canetti et al's GUC framework. In Canetti's UC framework, 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 [11]. 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.* $\mathsf{GEXEC}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathsf{GEXEC}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$.

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 [11]. 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.

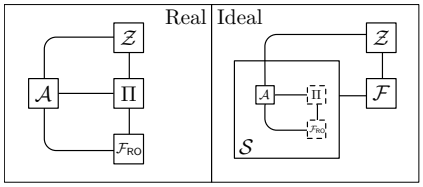
Definition 4. *Let the protocol Π be $\mathcal{G}$-subroutine respecting. We say a protocol Π EUC-realizes functionality $\mathcal{F}$ with respect to shared functionality $\mathcal{G}$, if for any PPT $\mathcal{G}$-externalized constrained environment $\mathcal{Z}$ and any PPT adversary $\mathcal{A}$ 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 [11], 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.

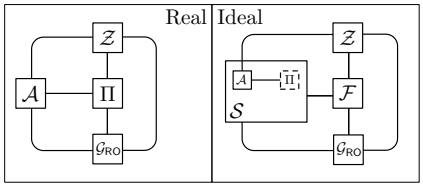
2.3 The Global Random Oracle Models

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 [14], which does not give any extra power to anyone; (ii) Global Observable Random Oracle (GORO) model[5] proposed by Canetti *et al.* in [14], which grants

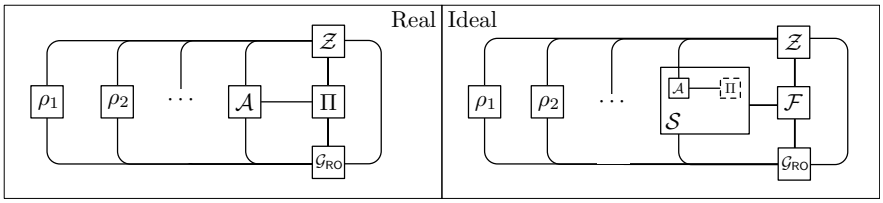
[5] In [9], Camenisch *et al.* used the notations Restricted Observable Global Random oracles (GroRO), Restricted Programmable Global Random Oracles (GrpRO) and



(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 $\mathcal{G}_{\mathsf{RO}}$-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.

Fig. 2. Comparison of Basic UC, GUC and EUC.

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 [9], which allows the simulator to program on unqueried points without being detected; (iv) Global Programmable and Observable Random Oracle (GPORO) model proposed by Camenisch *et al.* in [9], which provides both programmability and observability. We present the relation of these models in Figure 3, and the formal description of all the global random oracle models mentioned above 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 4(a), upon receiving $(\textsc{Query}, \mathsf{sid}, x)$ from any party, $\mathcal{G}_{\mathsf{sRO}}$ first checks if the query (sid, x) has been 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 "illegiti-

Restricted Observable and Programmable Global Random Oracles (GrpoRO). Here we adopt the notations GORO, GPRO and GPORO which skips the "r" for the sake of the simplicity as in [15].

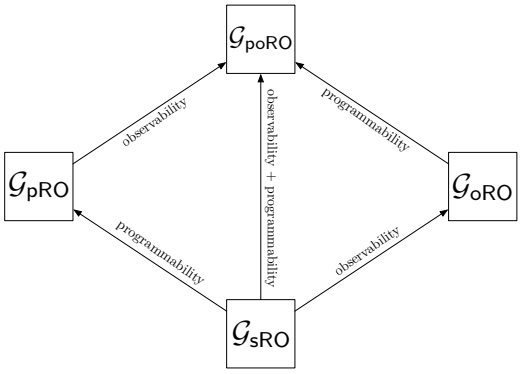


Fig. 3. Relation of the Global Random Oracle Models

mate" and potentially disclosed to the simulator. As depicted in Figure 4(b), the GORO functionality $\mathcal{G}_{\mathsf{oRO}}$ interacts with a list of ideal functionality programs $\bar{\mathcal{F}} = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$, where $\mathcal{F}_1, \ldots, \mathcal{F}_n$ are the protocol functionalities (e.g., commitment functionality, ZK functionality, etc.) that share the same global setup $\mathcal{G}_{\mathsf{oRO}}$. 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 4(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 4(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 [9].

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 [9]. More precisely, as depicted in Figure 5, the reduction

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

The functionality interacts with a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ 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 $P_i \in \mathcal{P}$ where $P_i = (\mathsf{pid}, \mathsf{sid})$, 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 parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ 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 4(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 parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ 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 4(a).
- **Program.** Upon receiving $(\textsc{Program}, \mathsf{sid}, x, v)$ with $v \in \{0,1\}^{\ell_{\mathsf{out}}(\lambda)}$ from $\mathcal{S}$:
  - 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 or $\mathcal{S}$:
  - If the input was given by $P_i = (\mathsf{pid}, \mathsf{sid})$ and $\mathsf{sid} \neq \mathsf{sid}'$, ignore this input.
  - If $(\mathsf{sid}', x) \in \mathsf{Prog}$, set $b := 1$; otherwise, set $b := 0$.
  - 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 parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ 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 4(b).
- **Program/IsPrograMed.** Same as $\mathcal{G}_{\mathsf{pRO}}$ depicted in Figure 4(c).

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

Fig. 4. The Global Random Oracle Models.

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 $\mathcal{G}$.

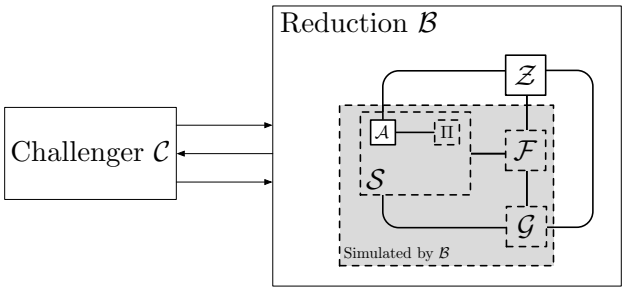


Fig. 5. In order to play against the external challenger $\mathcal{C}$, reduction algorithm $\mathcal{B}$ simulates everything (marked as gray) including the shared functionality $\mathcal{G}$, then starts the protocol Π with the real world adversary $\mathcal{A}$/environment $\mathcal{Z}$ by running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

2.4 SHVZK Protocols

A 3-round public coin Special Honest Verifier Zero-Knowledge (SHVZK) protocol $\Pi = \Pi.\{\mathsf{Move1}, \mathsf{Move2}, \mathsf{Move3}, \mathsf{Verify}, \mathsf{Sim}\}$ allows a prover to convince a verifier that a statement x is true with the aid of the witness w. In the first round, the prover computes and sends the first flow message $a := \mathsf{Move1}(x, w; r)$ using the statement-witness pair (x, w) and some random coin r. In the second round, the verifier samples and sends a uniformly random public coin challenge $e \leftarrow \mathsf{Move2}(1^\lambda)$. In the last round, the prover computes the response to the challenge $z := \mathsf{Move3}(x, w, e; r)$ using the statement-witness pair (x, w), challenge e and the random coin r. Finally the verifier accepts the statement x if and only if $\mathsf{Verify}(x, a, e, z) = 1$. We put the workflow of the SHVZK protocol in Figure 6. We often call (a, e, z) the transcript between the prover and the verifier.

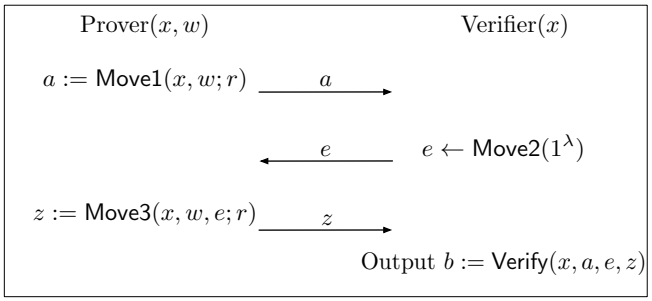


Fig. 6. The Workflow of the SHVZK Protocol

A SHVZK protocol should satisfy (i) perfect completeness, i.e. any honest prover who holds the witness w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$ can always make the verifier accept; (ii) k-special soundness, i.e. given any k distinct accepting transcripts, we can always extract the witness w; (iii) Special Honest Verifier Zero-Knowledge (SHVZK) property, i.e. given the challenge e ahead, there should be a PPT simulator algorithm Sim that takes the statement x, the challenge e and random coin r as input, and outputs the simulated (a, z) which is indistinguishable from the real one. The first property is easy to formalize. In order to formalize the k-special soundness and SHVZK, we consider the following experiments:

Experiment $\text{EXPT}^{\text{k-SS}}_{\mathcal{A},\Pi}(\lambda)$:
1. $\mathcal{A}$ outputs a statement x along with k transcripts $\{(a, e_i, z_i)\}_{i\in[k]}$.
2. If $e_i \neq e_j$ where $i \neq j$: extract the witness w' from $\{(a, e_i, z_i)\}_{i\in[k]}$
3. If $(x, w') \in \mathcal{R}_{\mathcal{L}}$, output 1; otherwise, output 0.

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

Experiment $\text{EXPT}^{\text{SHVZK}}_{\mathcal{A},\Pi}(\lambda)$:
1. $\mathcal{A}$ outputs a statement-witness pair (x, w) along with a challenge e.
2. If $(x, w) \in \mathcal{R}_{\mathcal{L}}$: select a random string r and a random bit $b \in \{0, 1\}$, and compute the following:
   (a) If $b = 0$: $a := \mathsf{Move1}(x, w; r)$; $z := \mathsf{Move3}(x, w, e; r)$.
   (b) If $b = 1$: $(a, z) := \mathsf{Sim}(x, e; r)$.
3. $\mathcal{A}$ is given (a, z) as input, and it outputs a guess bit $b' \in \{0, 1\}$.
4. If $b = b'$, output 1; otherwise, output 0.

Denote by $\mathbf{Adv}^{\text{SHVZK}}_{\mathcal{A},\Pi}(\lambda) := \left|\Pr[\text{EXPT}^{\text{SHVZK}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right|$ the advantage of $\mathcal{A}$.

Now we can formally define the SHVZK protocol.

Definition 5. *We say a protocol $\Pi = \Pi.\{\mathsf{Move1}, \mathsf{Move2}, \mathsf{Move3}, \mathsf{Verify}, \mathsf{Sim}\}$ is a SHVZK protocol if the following conditions hold:*

1. ***(Perfect Completeness)*** *For any $(x, w) \in \mathcal{R}_{\mathcal{L}}$, we say it is perfect complete if*

$$\Pr\left[\begin{array}{l} a := \mathsf{Move1}(x, w; r); e \leftarrow \mathsf{Move2}(1^\lambda); \\ z := \mathsf{Move3}(x, w, e; r) \end{array} : \mathsf{Verify}(x, a, e, z) = 1\right] = 1$$

2. ***(k-Special Soundness)*** *For any PPT adversary $\mathcal{A}$, we say it has k-special soundness where $k \in \mathbb{N}$ and $k \geq 2$, if there exists a negligible function negl such that $\mathbf{Adv}^{\text{k-SS}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.*
3. ***(Special Honest Verifier Zero-Knowledge)*** *We say it has SHVZK if there exists a PPT simulator Sim such that for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that $\mathbf{Adv}^{\text{SHVZK}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.*

2.5 Straight-Line Extractable NIWH Argument in the RO Model

Witness Hiding (WH) interactive proofs were introduced by Feige and Shamir in [21], and the Non-Interactive Witness-Hiding (NIWH) argument in the plain

model can be found in [33]. We here discuss the NIWH argument in the random oracle model. Note that, stronger security property such as *(straight-line) extractability* can now be achieved in the random oracle model: an extraction algorithm Ext could be constructed to extract the witness from a maliciously generated and accepting proof. More concretely, in an NIWH argument in the random oracle model $\Pi = \Pi.\{\mathsf{Prove}^{\mathcal{O}}, \mathsf{Verify}^{\mathcal{O}}, \mathsf{Ext}\}$, both the prover and the verifier are allowed to query the random oracle $\mathcal{O}$ at any moment, during the protocol execution. As in the plain model, the prover generates the proof π using the statement-witness pair (x, w) and a random string r and sends π to the verifier, and the verifier then verifies if the proof π is valid or not; the verifier outputs a bit b indicating the acceptance or rejection. Formally, the Prove and Verify algorithms in a NIWH argument in the RO model are described as follows:

- $\pi := \mathsf{Prove}^{\mathcal{O}}(x, w; r)$ takes input as a statement-witness pair (x, w) and a random string r, and it is allowed to query the random oracle $\mathcal{O}$. It outputs a proof π. When r is not important, we use $\mathsf{Prove}^{\mathcal{O}}(x, w)$ for simplicity.
- $b := \mathsf{Verify}^{\mathcal{O}}(x, \pi)$ takes input as a statement x and a proof π, and it is allowed to query the random oracle $\mathcal{O}$. It outputs a bit b indicating acceptance or rejection.

The straight-line extractable NIWH argument should satisfy the perfect completeness, computational soundness, witness hiding and straight-line extractability. The perfect completeness is trivial. The computational soundness means that any PPT prover cannot convince the verifier that a false statement is true with overwhelming probability. The last two properties are not to easy to formalize. We first talk about the witness hiding property: given the proof π generated by the prover, the verifier cannot compute any new witness that the verifier does not know before the interaction. In order to formally define the witness hiding property, we consider the following definition of hard instance ensembles [38].

Definition 6 (Hard Instance Ensembles). *Let $\mathcal{R}_{\mathcal{L}}$ 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 NP relation *$\mathcal{R}_{\mathcal{L}}$ if for any PPT $\mathcal{A}$ and any $x \in \mathcal{X}$, there exists a negligible function* negl *s.t.* $\Pr[(x, \mathcal{A}(x)) \in \mathcal{R}_{\mathcal{L}}] = \mathsf{negl}(\lambda)$.

Then we should consider the following experiment:

Experiment $\text{EXPT}^{\text{WH}}_{\mathcal{A},\Pi}(\lambda)$:
1. Select $(x, w) \in \mathcal{R}_{\mathcal{L}}$, and compute $\pi \leftarrow \mathsf{Prove}(x, w)$.
2. $\mathcal{A}$ is given (x, π) as input, and it outputs w'.
3. If $(x, w') \in \mathcal{R}_{\mathcal{L}}$, output 1; otherwise, output 0.

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

We now describe how to define the straight-line extractability property; note that our extractability definition is taken from that by Pass [38]. To enable the extractability, typically, the extraction algorithm Ext can be developed by simulating the random oracle for the prover and the verifier, and thus the algorithm

Ext has full control of the random oracle. In this paper, we consider a much more restricted random oracle, and the algorithm Ext is granted only with the *observability*; that is, Ext is allowed to see the query-answer list of the random oracle. For that reason, we write $\mathsf{Ext}^{\mathcal{O}}$ to indicate that, the extraction algorithm Ext does not have the full control of the random oracle, and is only granted to have the observability capability. With these notions above, we can formally define the straight-line extractable NIWH arguments in the RO model.

Definition 7. *Fix an NP relation $\mathcal{R}_{\mathcal{L}}$ whose associate language is $\mathcal{L}$. Consider a RO $\mathcal{O}$. We say a protocol $\Pi = \Pi.\{\mathsf{Prove}, \mathsf{Verify}\}$ is a NIWH argument for $\mathcal{R}_{\mathcal{L}}$ in the RO model if the following condition holds:*

1. ***(Perfect Completeness)*** *For any $(x, w) \in \mathcal{R}_{\mathcal{L}}$, we say it is perfect complete if*
$$\Pr[\pi \leftarrow \mathsf{Prove}^{\mathcal{O}}(x, w) : \mathsf{Verify}^{\mathcal{O}}(x, \pi) = 1] = 1$$
2. ***(Computational Soundness)*** *For any $x \notin \mathcal{L}$, we say it is computational sound if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that*
$$\Pr[\pi^* \leftarrow \mathcal{A}^{\mathcal{O}}(x) : \mathsf{Verify}^{\mathcal{O}}(x, \pi^*) = 1] \leq \mathsf{negl}(\lambda)$$
3. ***(Witness Hiding)*** *Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ be a hard instance ensemble $\mathcal{R}_{\mathcal{L}}$. We say it is witness hiding for $\mathcal{R}_{\mathcal{L}}$ under the instance ensemble $\mathcal{X}$ if for any PPT adversary $\mathcal{A}$ and any $(x, w) \in \mathcal{R}_{\mathcal{L}}$ with $x \in \mathcal{X}$, there exists a negligible function negl s.t. $\mathbf{Adv}^{\mathrm{WH}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$. We say it is witness hiding for $\mathcal{R}_{\mathcal{L}}$ if it is witness hiding under all hard-instance ensembles $\mathcal{X}$ for $\mathcal{R}_{\mathcal{L}}$.*
4. ***(Straight-line Extractability)*** *For any $x \in \mathcal{L}$, we say it is straight-line extractable if for any PPT adversary $\mathcal{A}$,*
$$\Pr\left[\begin{matrix}\pi^* \leftarrow \mathcal{A}^{\mathcal{O}}(x); b := \mathsf{Verify}^{\mathcal{O}}(x, \pi^*); \\ w^* \leftarrow \mathsf{Ext}^{\mathcal{O}}(x, \pi^*)\end{matrix} : b = 1 \ \wedge \ (x, w^*) \in \mathcal{R}_{\mathcal{L}}\right] \geq 1 - \mathsf{negl}(\lambda)$$

2.6 Equivocal Commitment

Typically, an equivocal commitment scheme $\Pi = \Pi.\{\mathsf{KeyGen}, \mathsf{KeyVer}, \mathsf{Commit}, \mathsf{ComVer}, \mathsf{EquCom}, \mathsf{Equiv}\}$ allows the committer to generate the commitment 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 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 the trapdoor td.
- $b := \mathsf{KeyVer}(\mathsf{ck}, \mathsf{td})$ takes input as a commitment key ck and a trapdoor td. It outputs a bit b indicating acceptance or rejection.
- $(c, d) := \mathsf{Commit}(\mathsf{ck}, m; r)$ 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.

- $b := \mathsf{ComVer}(\mathsf{ck}, c, d)$ takes input as a commitment key ck, and a commitment-opening pair (c, d). It outputs a bit b indicating acceptance or rejection.
- $(\tilde{c}, \mathsf{st}) := \mathsf{EquCom}(\mathsf{ck}, \mathsf{td}; r)$ takes input as a commitment key ck, a 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.
- $\tilde{d} := \mathsf{Equiv}(\mathsf{ck}, \mathsf{td}, \tilde{c}, \mathsf{st}, \tilde{m})$ takes input as a commitment key ck, a 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 requires the following properties: perfect correctness, perfect hiding, computational binding and equivocation. Perfect correctness means that the honest committer can always make the receiver accept. Perfect hiding means that the commitment reveals nothing about the message.

Experiment $\text{EXPT}^{\text{hiding}}_{\mathcal{A},\Pi}(\lambda)$:

1. Run $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2. $\mathcal{A}$ is given ck as input, and it outputs two distinct messages m_0, m_1.
3. Select a random bit $b \in \{0, 1\}$, and compute $c \leftarrow \mathsf{Commit}(\mathsf{ck}, m_b)$.
4. $\mathcal{A}$ is given m_b as input, and it outputs a bit b'.
5. If $b' = b$, output 1; otherwise, output 0.

Denote by $\mathbf{Adv}^{\text{hiding}}_{\mathcal{A},\Pi}(\lambda) := \left|\Pr[\text{EXPT}^{\text{hiding}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right|$ the advantage of $\mathcal{A}$.

Computational binding means that it is infeasible for the PPT committer to output the commitment c that can be opened in two different ways.

Experiment $\text{EXPT}^{\text{binding}}_{\mathcal{A},\Pi}(\lambda)$:

1. Run $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2. $\mathcal{A}$ is given ck as input, and it outputs (c, d_0, d_1).
3. If $d_0 \neq d_1$ and $\mathsf{ComVer}(\mathsf{ck}, c, d_0) = \mathsf{ComVer}(\mathsf{ck}, c, d_1) = 1$ holds, output 1; otherwise, output 0.

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

Equivocation means that given the trapdoor td, one can open a previously constructed commitment c of message m to other message $\tilde{m} \neq m$.

Experiment $\text{EXPT}^{\text{equivocal}}_{\mathcal{A},\Pi}(\lambda)$:

1. $\mathcal{A}$ is given 1^λ as input, and it outputs $(\mathsf{ck}, \mathsf{td}, m)$.
2. If $\mathsf{KeyVer}(\mathsf{ck}, \mathsf{td}) = 1$: select a random string r and a random bit $b \in \{0, 1\}$, and compute the following:
   (a) If $b = 0$: invoke $(c, d) := \mathsf{Commit}(\mathsf{ck}, m; r)$.
   (b) If $b = 1$: invoke $(c, \mathsf{st}) := \mathsf{EquCom}(\mathsf{ck}, \mathsf{td}; r); d := \mathsf{Equiv}(\mathsf{ck}, \mathsf{td}, \tilde{c}, \mathsf{st}, m)$.
3. $\mathcal{A}$ is given (c, d) as input, and it outputs a bit b'.
4. If $b = b'$, output 1; otherwise, output 0.

Denote by $\mathbf{Adv}^{\text{equivocal}}_{\mathcal{A},\Pi}(\lambda) := \left|\Pr[\text{EXPT}^{\text{equivocal}}_{\mathcal{A},\Pi}(\lambda) = 1] - \frac{1}{2}\right|$ the advantage of $\mathcal{A}$.

Now we can formally define the equivocal commitment, and it should satisfy the following definition:

Definition 8. *We say a scheme* $\Pi = \Pi.\{\mathsf{KeyGen}, \mathsf{KeyVer}, \mathsf{Commit}, \mathsf{ComVer}, \mathsf{EquCom}, \mathsf{Equiv}\}$ *is an equivocal commitment if the following conditions hold:*

1. ***(Perfect Correctness)*** *For any message* m*, we say it is perfect correct if*

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

2. ***(Perfect Hiding)*** *We say it is perfect hiding if for any adversary* $\mathcal{A}$ *s.t.* $\mathbf{Adv}^{\text{hiding}}_{\mathcal{A},\Pi}(\lambda) = 0$.
3. ***(Computational Binding)*** *We say it is computational binding if for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *s.t.* $\mathbf{Adv}^{\text{binding}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.
4. ***(Equivocation)*** *We say it is equivocal if for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* negl *s.t.* $\mathbf{Adv}^{\text{equivocal}}_{\mathcal{A},\Pi}(\lambda) \leq \mathsf{negl}(\lambda)$.

2.7 "MPC-in-the-Head" Paradigm

In [30], Ishai *et al.* proposed the famous "MPC-in-the-head" paradigm from which we can construct a SHVZK protocol using the MPC protocol. Before introducing the details of the paradigm, we have to define the MPC protocol.

Consider a function $f : (\{0,1\}^\lambda)^{n+1} \to \{0,1\}^\lambda$. We let $P_1, \ldots, P_n$ be n parties modeled as PPT interactive machines. Assume that 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 (e.g. encrypted channels or OT channels) in the synchronous model. 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 denote by $\mathsf{view}_i(x, w_i)$ the view of P_i, which is 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}. Each secure 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 in $\mathsf{view}_i(x, w_i)$ are identical to the incoming messages 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; while in the general case, the output of P_i can be different from each other.

In this work, we only consider security of MPC protocols in the semi-honest model. 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 $j \neq i$.

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} 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 notations, we have the following definition.

Definition 9. *We say an n-party protocol Π_{MPC} realizes f in the semi-honest model, if the following conditions hold:*

1. ***(Perfect 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 $\forall i \in [n] : \Pr[y = \Pi_{f,i}(\mathsf{view}_i(x, w_i))] = 1$.*
2. ***(t-Privacy)** Let $1 \leq t < n$. We say Π_{MPC} realizes f with 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*

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

Now we can introduce the "MPC-in-the-head" paradigm. Let f be the following $(n+1)$-argument function corresponding to an NP relation $\mathcal{R}_{\mathcal{L}}$, that is, $f(x, w_1, \ldots, w_n) = \mathcal{R}_{\mathcal{L}}(x, w_1 \oplus \cdots \oplus w_n)$. Here 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. 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 accepts the statement if and only if (i) the commitments are correctly opened and (ii) the opened views are consistent with each other. See more details in [30].

3 Impossibility in the GORO Model

In this section, we show that it is impossible to construct 2-round GUC-secure commitment protocols (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 [14] in Figure 7. The main difference with the traditional commitment functionality is that in $\mathcal{F}_{\mathsf{tCOM}}$, the simulator can request the list of the illegitimate queries from $\mathcal{F}_{\mathsf{tCOM}}$. If we use the traditional commitment functionality which has no such power in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world, the simulator will have no advantage over others at all. This is one of the reasons why transferable ideal functionalities were designed in the presence of the $\mathcal{G}_{\mathsf{oRO}}$ model, and we refer interesting readers to see more discussions in [14].

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 $\mathcal{S}$.

Fig. 7. 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 advantages 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 7 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 protocol 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, $\mathcal{Z}$ lets 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 $\tilde{\psi}$ without knowing b in the committing phase, where $\tilde{\psi}$ is computational indistinguishable from the real commitment ψ; later in the opening phase, $\mathcal{S}$ obtains b from $\mathcal{F}_{\mathsf{tCOM}}$ and needs to open the previously sent commitment $\tilde{\psi}$ to b. For notation convenience, we write $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ to split the simulator algorithm in two phases: (i) $\mathcal{S}_1$ works in the committing phase, and it needs to output an equivocal commitment $\tilde{\psi}$ without knowing b; (ii) while $\mathcal{S}_2$ works in the opening phase, and upon receiving the message b from $\mathcal{F}_{\mathsf{tCOM}}$, it needs to output the opening message r such that (b, r) correctly opens the previously sent commitment $\tilde{\psi}$.

We first describe the simulation strategy in the committing phase. Recall that, the main advantage of the simulator over the others is that it can obtain illegitimate queries of R^*. More precisely, $\mathcal{S}_1$ 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 $\mathcal{S}_1$ also can query $\mathcal{G}_{\mathsf{oRO}}$ just like normal parties. In order to describe the process of querying to $\mathcal{G}_{\mathsf{oRO}}$, we denote by $\mathcal{G}^*_{\mathsf{oRO}}$ the simplified version of the $\mathcal{G}_{\mathsf{oRO}}$, that is, the $\mathcal{G}_{\mathsf{oRO}}$ with only the QUERY interface. We write $\mathcal{S}_1^{\mathcal{G}^*_{\mathsf{oRO}}}$ to denote the event that $\mathcal{S}_1$ has the query access to $\mathcal{G}_{\mathsf{oRO}}$ and can continuously query to $\mathcal{G}_{\mathsf{oRO}}$. With above notations, we will write $\mathcal{S}_1^{\mathcal{G}^*_{\mathsf{oRO}}}(\mathsf{sid}_1, \mathcal{Q}_{\mathsf{sid}_1})$ to denote the output (i.e., the equivocal commitment $\tilde{\psi}$ and the state st) produced by $\mathcal{S}_1$ after querying to $\mathcal{G}_{\mathsf{oRO}}$, when running on the the illegitimate queries $\mathcal{Q}_{\mathsf{sid}_1}$ sent by R^*. We note that, $\mathcal{S}_1$ should be able to handle any PPT environment $\mathcal{Z}$. Consider such a case where $\mathcal{Z}$ instructs R^* to delay all its $\mathcal{G}_{\mathsf{oRO}}$ queries until it receives the opening message. In this case, $\mathcal{S}_1$ finds nothing sent by R^* in $\mathcal{Q}_{\mathsf{sid}_1}$, but should still be able to produce the equivocal commitment $\tilde{\psi}$. In other words, the algorithm $(\tilde{\psi}, \mathsf{st}) \leftarrow \mathcal{S}_1^{\mathcal{G}^*_{\mathsf{oRO}}}(\mathsf{sid}_1, \mathcal{Q}_{\mathsf{sid}_1})$ still works when $\mathcal{Q}_{\mathsf{sid}_1} = \emptyset$, where $\emptyset$ is an empty set; otherwise, the environment $\mathcal{Z}$ will find the distinction. We note that, the algorithm $\mathcal{S}_1^{\mathcal{G}^*_{\mathsf{oRO}}}(\mathsf{sid}_1, \emptyset)$, i.e. we replace the $\mathcal{Q}_{\mathsf{sid}_1}$ with the empty set $\emptyset$, can be run by any party, since the algorithm only makes use of the QUERY interface and anyone can query to $\mathcal{G}_{\mathsf{oRO}}$. Now let us turn to the opening phase. Analogously, we can write $r \leftarrow \mathcal{S}_2^{\mathcal{G}^*_{\mathsf{oRO}}}(\mathsf{sid}_1, \tilde{\psi}, \mathsf{st}, b, \emptyset)$ to denote the event where $\mathcal{S}_2$ can still open $\tilde{\psi}$ to the value b and the corresponding opening message r after querying to $\mathcal{G}_{\mathsf{oRO}}$, even if there is noting sent by R^* in the list of the illegitimate queries (i.e. $\mathcal{Q}_{\mathsf{sid}_1} = \emptyset$). We note that, even if we switch to a session with a different SID, both $\mathcal{S}_1^{\mathcal{G}^*_{\mathsf{oRO}}}(\mathsf{sid}_1, \emptyset)$ and $\mathcal{S}_2^{\mathcal{G}^*_{\mathsf{oRO}}}(\mathsf{sid}_1, \tilde{\psi}, \mathsf{st}, b, \emptyset)$ still work as long as the appropriate inputs are provided.

In the following, we show that the existence of the simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ 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 $(\tilde{\psi}, \mathsf{st}) \leftarrow \mathcal{S}_1^{\mathcal{G}^*_{\mathsf{oRO}}}(\mathsf{sid}_2, \emptyset)$ and send $\tilde{\psi}$ to R. When R outputs $(\textsc{Receipt}, \mathsf{sid}_2, C, R)$, $\mathcal{Z}'$ instructs C^* to run the algorithm $r \leftarrow \mathcal{S}_2^{\mathcal{G}^*_{\mathsf{oRO}}}(\mathsf{sid}_2, \tilde{\psi}, \mathsf{st}, b, \emptyset)$ 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 $\tilde{\psi}$ 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 $\tilde{\psi}$ 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.

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 captures 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.

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 in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world 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 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 $y := \mathsf{oRO}(x)$. We assume ideal private and authenticated channels for all communications. Formally, we present our protocol Π_{tCOM} in Figure 8 and prove the security through Theorem 2.

Protocol Π_{tCOM}

Primitives: Straight-line extractable NIWH argument in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world $\Pi_{\mathsf{NIWH}} = \Pi_{\mathsf{NIWH}}.\{\mathsf{Prove}^{\mathcal{G}_{\mathsf{oRO}}}, \mathsf{Verify}^{\mathcal{G}_{\mathsf{oRO}}}, \mathsf{Ext}^{\mathcal{G}_{\mathsf{oRO}}}\}$, non-interactive equivocal commitment $\Pi_{\mathsf{ECom}} = \Pi_{\mathsf{ECom}}.\{\mathsf{KeyGen}, \mathsf{KeyVer}, \mathsf{Commit}, \mathsf{ComVer}, \mathsf{EquCom}, \mathsf{Equiv}\}$.
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 parameters of the commitment by invoking $(\mathsf{ck}, \mathsf{td}) \leftarrow \Pi_{\mathsf{ECom}}.\mathsf{KeyGen}(1^\lambda)$.
  - Compute the straight-line extractable NIWH proof by invoking $\pi \leftarrow \Pi_{\mathsf{NIWH}}.\mathsf{Prove}^{\mathcal{G}_{\mathsf{oRO}}}(\mathsf{ck}, \mathsf{td})$ for proving the knowledge of td. Send (ck, π) to C.
- Round 2: C works as follows:
  - Abort if $\Pi_{\mathsf{NIWH}}.\mathsf{Verify}^{\mathcal{G}_{\mathsf{oRO}}}(\mathsf{ck}, \pi) = 0$.
  - Commit to the message m by invoking $(c_1, d_1) \leftarrow \Pi_{\mathsf{ECom}}.\mathsf{Commit}(\mathsf{ck}, m)$.
  - Compute $h := \mathsf{oRO}(\mathsf{sid}, \text{`}C\text{'}||m||d_1||r)$ where $r \leftarrow \{0,1\}^\lambda$.
  - Commit to h by invoking $(c_2, d_2) \leftarrow \Pi_{\mathsf{ECom}}.\mathsf{Commit}(\mathsf{ck}, h)$. 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 $h := \mathsf{oRO}(\mathsf{sid}, \text{`}C\text{'}||m||d_1||r)$, and accepts m if and only if $\Pi_{\mathsf{ECom}}.\mathsf{ComVer}(\mathsf{ck}, c_1, d_1) = \Pi_{\mathsf{ECom}}.\mathsf{ComVer}(\mathsf{ck}, c_2, d_2) = 1$ holds.

Fig. 8. Protocol Π_{tCOM} in the $\mathcal{G}_{\mathsf{oRO}}$ Hybrid World

Theorem 2. *Assume Π_{NIWH} is a straight-line extractable NIWH argument in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world. Assume Π_{ECom} is an equivocal commitment scheme. Then the protocol Π_{tCOM} described in Figure 8 GUC-realizes the functionality $\mathcal{F}_{\mathsf{tCOM}}$ depicted in Figure 7 in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world against static malicious corruption.*

Proof. We leave the proof in the full version.

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.

SHVZK protocols from "MPC-in-the-head". We here 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 introduced in Section 2.7.

Note that our construction requires an SHVZK protocol with 2-special soundness (, which we will explain the necessity later in Section 4.2); unfortunately, to the best of our knowledge, none of the followups [25,16,1,31,19] since the original work of [30], can lead to a 2-special sound protocol under 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.

Our starting point: [31]. We start with the 5-round SHVZK protocol proposed by Katz *et al.* in [31] 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 executions, 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 checks $\mathcal{R}_{\mathcal{L}}(x,w)=1$ 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 states of each challenged party following the preprocessing phase as well as its views 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 states from the preprocessing phase) that yields the output 1.

In [31], 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 recall the formal 3-round SHVZK protocol of Katz *et al.* in the full version of our paper. We emphasize that the 3-round SHVZK protocol proposed by Katz *et al.* cannot be 2-special sound, and we argue this through 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 described in [31]:*

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

Proof. We leave the proof in the full version.

Our protocol construction. Our key observation is that we can compress the above 5-round protocol into a 3-round one by applying the Fiat-Shamir transformation [22] 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 [31], our approach needs additional RO assumptions but it is an SHVZK protocol with 2-special sound.

Protocol Π_{SHVZK}

Primitives: n-party MPC protocol Π_{MPC} which realizes f with $(n-1)$-privacy in the preprocessing model, where $f(x, w_1, \ldots, w_n) = \mathcal{R}_{\mathcal{L}}(x, w_1 \oplus \cdots \oplus w_n)$.
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 have a common input x. P has a private input w s.t. $\mathcal{R}_{\mathcal{L}}(x, w) = 1$.

– $\mathsf{Move1}(x, w; r)$:
  • 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 $r_{i,j,k} \leftarrow \{0,1\}^{\lambda}$ and commit to the states, i.e. compute state-commitments $\mathsf{com}_{i,j,k} := \mathsf{oRO}_1(\mathsf{sid}, \mathsf{state}_{i,j,k} || r_{i,j,k})$.
  • Compute $(c_1, \ldots, c_\lambda) := \mathsf{oRO}_2(\mathsf{sid}, \{\mathsf{com}_{i,j,k}\}_{i\in[\lambda], j\in[m], k\in[n]})$, where $c_i \in [m]$.
  • 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 $\tilde{r}_{i,k} \leftarrow \{0,1\}^{\lambda}$ and commit to the view of each party, i.e. compute view-commitments $\widetilde{\mathsf{com}}_{i,k} := \mathsf{oRO}_1(\mathsf{sid}, \mathsf{view}_{i,k}(x, w_k) || \tilde{r}_{i,k})$.
  • Send $a := (\{\mathsf{com}_{i,j,k}, \widetilde{\mathsf{com}}_{i,k}\}_{i\in[\lambda], j\in[m], k\in[n]}, \{\mathsf{state}_{i,j,k}, r_{i,j,k}\}_{i\in[\lambda], j\in[m]\setminus\{c_i\}, k\in[n]})$.
– $\mathsf{Move2}(1^{\lambda})$: Send $e := (p_1, \ldots, p_\lambda)$, where $p_i \in [n]$ and p_i is uniformly random.
– $\mathsf{Move3}(x, w, e; r)$: Send $z := (\{\mathsf{view}_{i,k}(x, w_k), \tilde{r}_{i,k}, \mathsf{state}_{i,c_i,k}, r_{i,c_i,k}\}_{i\in[\lambda], k\in[n]\setminus\{p_i\}})$.
– $\mathsf{Verify}(x, a, e, z)$: Output 1 if and only if the following checks pass:
  • Check the commitments are opened correctly:
    ∗ For $i \in [\lambda], j \in [m] \setminus \{c_i\}, k \in [n]$: check $\mathsf{com}_{i,j,k} = \mathsf{oRO}_1(\mathsf{sid}, \mathsf{state}_{i,j,k} || r_{i,j,k})$ holds.
    ∗ For $i \in [\lambda], k \in [n] \setminus \{p_i\}$: check $\mathsf{com}_{i,c_i,k} = \mathsf{oRO}_1(\mathsf{sid}, \mathsf{state}_{i,c_i,k} || r_{i,c_i,k})$ and $\widetilde{\mathsf{com}}_{i,k} = \mathsf{oRO}_1(\mathsf{sid}, \mathsf{view}_{i,k}(x, w_k) || \tilde{r}_{i,k})$ hold.
  • Check the correctness of the executions of the preprocessing phase:
    ∗ Compute $(c_1, \ldots, c_\lambda) := \mathsf{oRO}_2(\mathsf{sid}, \{\mathsf{com}_{i,j,k}\}_{i\in[\lambda], j\in[m], k\in[n]})$.
    ∗ For $i \in [\lambda], j \in [m] \setminus \{c_i\}$: check $\{\mathsf{state}_{i,j,k}\}_{k\in[n]}$ are well-formed.
  • Check the consistency between the opened views:
    ∗ For $i \in [\lambda], k \in [n] \setminus \{p_i\}$: 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 $\{\mathsf{view}_{i,k}(x, w_k)\}_{k\in[n]\setminus\{p_i\}}$ are consistent with each other.

Fig. 9. Protocol Π_{SHVZK} in the $\mathcal{G}_{\mathsf{oRO}}$ Hybrid World

Let Π_{MPC} be the n-party MPC protocol which realizes f with $(n-1)$-privacy in the preprocessing model, where $f(x, w_1, \cdots, w_n) = \mathcal{R}_{\mathcal{L}}(x, w_1 \oplus \cdots \oplus w_n)$. Let $\mathsf{Preprocess}$ be the preprocessing algorithm that takes a λ-bit random string seed as input, and outputs the states $\{\mathsf{state}\}_{i\in[n]}$ which are used for the computation later (cf. [31] for details). We use the $\mathcal{G}_{\mathsf{oRO}}$ to instantiate the statically binding commitment (i.e., to commit msg with random string 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 $y := \mathsf{oRO}_i(x)$ 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 9 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 $f(x, w_1, \cdots, w_n) = \mathcal{R}_{\mathcal{L}}(x, w_1 \oplus \cdots \oplus w_n)$. Then the protocol Π_{SHVZK} depicted in Figure 9 is a SHVZK protocol that satisfies perfect completeness, 2-special soundness, perfect SHVZK.*

Proof. We leave the proof in the full version.

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 $\mathcal{R}_{\mathcal{L}}$ be a hard NP relation. The receiver selects $(x, w) \in \mathcal{R}_{\mathcal{L}}$, 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 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 [18,35].

Let g be a one-way function; note that, due to the limit of space, we do not give the formal definition of one-way function, and we refer readers to see the definition in [32]. Formally, we present our non-interactive equivocal commitment in Figure 10 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. We instantiate the NP relation with one-way function, i.e. $\mathcal{R}_1 = \{(y, \mathsf{seed}) \mid y = g(\mathsf{seed})\}$ where (y, seed) is the statement-witness pair and g is a one-way function. If we use our SHVZK protocol Π_{SHVZK} depicted in Figure 9 as the building block, then we can obtain a perfect hiding non-interactive equivocal commitment scheme via only Minicrypt assumptions in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world.

Theorem 4. *Assume Π_{SHVZK} is a 2-special sound SHVZK protocol. Assume g is a one-way function. Then Π_{ECom} depicted in Figure 10 is an equivocal commitment that satisfies perfect correctness, perfect hiding, computational binding and perfect equivocation.*

Proof. We leave the proof in the full version.

Scheme Π_{ECom}

Primitives: SHVZK protocol $\Pi_{\mathsf{SHVZK}} = \Pi_{\mathsf{SHVZK}}.\{\mathsf{Move1}, \mathsf{Move2}, \mathsf{Move3}, \mathsf{Verify}, \mathsf{Sim}\}$ and one-way function g.

- $\mathsf{KeyGen}(1^\lambda)$: Select a random string $\mathsf{seed} \leftarrow \{0,1\}^\lambda$, compute $y := g(\mathsf{seed})$, and output $\mathsf{ck} := y, \mathsf{td} := \mathsf{seed}$.
- $\mathsf{KeyVer}(\mathsf{ck}, \mathsf{td})$: Check if $\mathsf{ck} = g(\mathsf{td})$ holds. If so, output 1; otherwise, output 0.
- $\mathsf{Commit}(\mathsf{ck}, m)$: Select a random string $r \leftarrow \{0,1\}^\lambda$, invoke $(a, z) := \Pi_{\mathsf{SHVZK}}.\mathsf{Sim}(\mathsf{ck}, m; r)$, and output $c := a, d := (m, z)$.
- $\mathsf{ComVer}(\mathsf{ck}, c, d)$: Check if $\Pi_{\mathsf{SHVZK}}.\mathsf{Verify}(\mathsf{ck}, c, m, z) = 1$ holds. If so, output 1; otherwise, output 0.
- $\mathsf{EquCom}(\mathsf{ck}, \mathsf{td})$: Select a random string $s \leftarrow \{0,1\}^\lambda$, invoke $\tilde{a} := \Pi_{\mathsf{SHVZK}}.\mathsf{Move1}(\mathsf{ck}, \mathsf{td}; s)$, and output $\tilde{c} := \tilde{a}, \mathsf{st} := s$.
- $\mathsf{Equiv}(\mathsf{ck}, \mathsf{td}, \tilde{c}, \mathsf{st}, \tilde{m})$: Invoke $\tilde{z} := \Pi_{\mathsf{SHVZK}}.\mathsf{Move3}(\mathsf{ck}, \mathsf{td}, \tilde{m}; \mathsf{st})$, and output $\tilde{d} := \tilde{z}$.

Fig. 10. Scheme Π_{ECom} Based on One-Way Function

Straight-line extractable NIWH argument. We construct the straight-line extractable NIWH argument in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world using the technique described in [38]. We here describe the high-level description and the details can be found in the full version of our paper. 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 [22]. The straight-line extractablity relies on the observability provided by $\mathcal{G}_{\mathsf{oRO}}$ and 2-special soundness.

Theorem 5 ([38]). *Assume there is a 2-special sound SHVZK protocol, then there exists a straight-line extractable NIWH argument in the $\mathcal{G}_{\mathsf{oRO}}$ hybrid world.*

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 under 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 under Minicrypt assumptions 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 [11] 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 [9].

Among all the global random oracle models depicted in Figure 4, 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 $\mathcal{G}_{\mathsf{pRO}}$ 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 the simulator still succeeds in extracting the committed value from the commitment message, then we can use such a simulator to break the hiding property of the commitment scheme 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 the full version of our paper.

Given this lower bound in the $\mathcal{G}_{\mathsf{pRO}}$, we find that the 3-round (2 rounds for the committing phase, 1 round for the opening phase) GUC-secure commitment protocol proposed in [9] is round-optimal. But their construction relies on CDH assumption which lives in Cryptomania world. Unfortunately, we find it extremely hard to construct a round-optimal GUC-secure commitment protocol under only Minicrypt assumptions in the $\mathcal{G}_{\mathsf{pRO}}$ hybrid world, so we leave it as an open question.

References

1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In ACM CCS 2017. pp. 2087–2104. ACM Press (2017).
2. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In FOCS 2004. pp. 186–195. IEEE Computer Society Press (2004).

3. Baum, C., David, B., Dowsley, R.: Insured MPC: Efficient secure computation with financial penalties. In FC 2020. LNCS, vol. 12059, pp. 404–420. Springer (2020).
4. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In ACM CCS 1993. pp. 62–73. ACM Press (1993).
5. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In Crypto 2019, Part III. LNCS, vol. 11694, pp. 701–732. Springer (2019).
6. Branco, P.: 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).
7. Branco, P., Goulão, M., Mateus, P.: UC-commitment schemes with phase-adaptive security from trapdoor functions. Cryptology ePrint Archive, Report 2019/529 (2019), https://eprint.iacr.org/2019/529.
8. Byali, M., Patra, A., Ravi, D., Sarkar, P.: 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.
9. Camenisch, J., Drijvers, M., Gagliardoni, T., Lehmann, A., Neven, G.: The wonderful world of global random oracles. In Eurocrypt 2018, Part I. LNCS, vol. 10820, pp. 280–312. Springer (2018).
10. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In FOCS 2001. pp. 136–145. IEEE Computer Society Press (2001).
11. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer (2007).
12. Canetti, R., Fischlin, M.: Universally composable commitments. In Crypto 2001. LNCS, vol. 2139, pp. 19–40. Springer (2001).
13. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited (preliminary version). In ACM STOC 1998. pp. 209–218. ACM Press (1998).
14. Canetti, R., Jain, A., Scafuro, A.: Practical UC security with a global random oracle. In ACM CCS 2014. pp. 597–608. ACM Press (2014).
15. Canetti, R., Sarkar, P., Wang, X.: Efficient and round-optimal oblivious transfer and commitment with adaptive security. In Asiacrypt 2020, Part III. LNCS, vol. 12493, pp. 277–308. Springer (2020).
16. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In ACM CCS 2017. pp. 1825–1842. ACM Press (2017).
17. Chiesa, A., Ojha, D., Spooner, N.: Fractal: Post-quantum and transparent recursive proofs from holography. In Eurocrypt 2020, Part I. LNCS, vol. 12105, pp. 769–793. Springer (2020).
18. Damgård, I.: On Σ-protocols. https://www.cs.au.dk/~ivan/Sigma.pdf.
19. de Saint Guilhem, C.D., Orsini, E., Tanguy, T.: Limbo: Efficient zero-knowledge MPCitH-based arguments. In ACM CCS 2021. pp. 3022–3036. ACM Press (2021).
20. Dodis, Y., Shoup, V., Walfish, S.: Efficient constructions of composable commitments and zero-knowledge proofs. In Crypto 2008. LNCS, vol. 5157, pp. 515–535. Springer (2008).
21. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In ACM STOC 1990. pp. 416–426. ACM Press (1990).
22. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In CRYPTO'1986. LNCS, vol. 263, pp. 186–194. Springer (1987).

23. Ganesh, C., Kondi, Y., Patra, A., Sarkar, P.: Efficient adaptively secure zero-knowledge from garbled circuits. In PKC 2018, Part II. LNCS, vol. 10770, pp. 499–529. Springer (2018).
24. Garg, S., Ishai, Y., Srinivasan, A.: Two-round MPC: Information-theoretic and black-box. In TCC 2018, Part I. LNCS, vol. 11239, pp. 123–151. Springer (2018).
25. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster zero-knowledge for Boolean circuits. In USENIX Security 2016. pp. 1069–1083. USENIX Association (2016)
26. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In ACM STOC 1987. pp. 218–229. ACM Press (1987).
27. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing **18**(1), 186–208 (1989)
28. Hofheinz, D., Müller-Quade, J.: Universally composable commitments using random oracles. In TCC 2004. LNCS, vol. 2951, pp. 58–76. Springer (2004).
29. Impagliazzo, R.: A personal view of average-case complexity. In SCT 1995. pp. 134–147. IEEE (1995)
30. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In ACM STOC 2007. pp. 21–30. ACM Press (2007).
31. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In ACM CCS 2018. pp. 525–537. ACM Press (2018).
32. Katz, J., Lindell, Y.: Introduction to modern cryptography. CRC press (2020)
33. Kuykendall, B., Zhandry, M.: Towards non-interactive witness hiding. In TCC 2020, Part I. LNCS, vol. 12550, pp. 627–656. Springer (2020).
34. Lysyanskaya, A., Rosenbloom, L.N.: Universally composable sigma-protocols in the global random-oracle model. Cryptology ePrint Archive, Paper 2022/290 (2022), https://eprint.iacr.org/2022/290.
35. MacKenzie, P.D., Yang, K.: On simulation-sound trapdoor commitments. In Eurocrypt 2004. LNCS, vol. 3027, pp. 382–400. Springer (2004).
36. Masny, D., Rindal, P.: Endemic oblivious transfer. In ACM CCS 2019. pp. 309–326. ACM Press (2019).
37. Mohassel, P., Rosulek, M., Scafuro, A.: Sublinear zero-knowledge arguments for RAM programs. In Eurocrypt 2017, Part I. LNCS, vol. 10210, pp. 501–531. Springer (2017).
38. Pass, R.: On deniability in the common reference string and random oracle model. In Crypto 2003. LNCS, vol. 2729, pp. 316–337. Springer (2003).
39. Yao, A.C.C.: Protocols for secure computations (extended abstract). In FOCS 1982. pp. 160–164. IEEE Computer Society Press (1982)