Black-box Constructions of Composable Protocols without Set-Up

Huijia Lin* Rafael Pass†

Abstract

We present the first *black-box* construction of a secure multi-party computation protocol that satisfies a meaningful notion of *concurrent security* in the plain model (without any set-up, and without assuming an honest majority). Moreover, our protocol relies on the minimal assumption of the existence of a semi-honest OT protocol, and our security notion "UC with super-polynomial helpers" (Canetti et al, STOC'10) is closed under universal composition, and implies "super-polynomial-time simulation".

*MIT and Boston University, E-Mail: huijia@csail.mit.edu.

†Cornell University, E-Mail: rafael@cs.cornell.edu. Pass is supported in part by a Microsoft New Faculty Fellowship, NSF CAREER Award CCF-0746990, AFOSR Award FA9550-08-1-0197, BSF Grant 2006317 and I3P grant 2006CS-001-0000001-02.

1 Introduction

The notion of *secure multi-party computation* allows m mutually distrustful parties to securely compute (or, *realize*) a functionality $f(\bar{x})$ of their corresponding private inputs $\bar{x} = x_1, ..., x_m$, such that party P_i receives the i^{th} component of $f(\bar{x})$. Loosely speaking, the security requirements are that the output of each party is distributed according to the prescribed functionality—this is called *correctness*—and that even malicious parties learn nothing more from the protocol than their prescribed output—this is called *privacy*. These properties should hold even in case that an arbitrary subset of the parties maliciously deviates from the protocol.

Soon after the concept was proposed [Yao86], general constructions were developed that appeared to satisfy the intuitive correctness and secrecy for practically any multi-party functionality [Yao86, GMW87]. These constructions require only authenticated communication and can use any enhanced trapdoor permutation. However, definitions that capture the security properties of secure multi-party computation protocols (and, in fact, of secure cryptographic protocols in general) took more time to develop. Here, the *simulation paradigm* emerged as a natural approach: Originally developed for capturing the security of encryption and then extended to Zero-Knowledge [GM84, GMR89]. The idea is to say that a protocol π securely realizes f if running π *"emulates"* an idealized process where all parties secretly provide inputs to an imaginary trusted party that computes f and returns the outputs to the parties; more precisely, any "harm" done by a polynomial-time adversary in the real execution of π, could have been done even by a polynomial-time adversary (called a *simulator*) in the ideal process. The simulation paradigm provides strong security guarantees: It ensures that running the protocols is "as good as" having a trusted third party computing the functionality for the players, and an adversary participating in the real execution of the protocols does not gain any "computational advantage" over the simulator in the ideal process (except from polynomial time advantage). We call this definition *basic security.*

The original setting in which secure multi-party protocols were investigated, however, only allowed the execution of a single instance of the protocol at a time; this is the so called stand-alone setting. A more realistic setting, is one which allows the concurrent execution of protocols. In the concurrent setting, many protocols are executed at the same time. This setting presents a new risk of a "coordinated attack" in which an adversary interleaves many different executions of a protocol and chooses its messages in each instance based on other partial executions of the protocol. To prevent coordinated attacks, we require the following basic security guarantee:

> **Concurrent Security:** The security properties, correctness and privacy, of the *analyzed protocol* should remain valid even when if multiple instance of the protocol are concurrently executed in a potentially unknown environment.

Another natural desideratum is the capability of supporting modular design of secure protocols.

> **Modular analysis:** The notion of security should support designing composite protocols in a modular way, while preserving security. That is, there should be a way to deduce security properties of the overall protocol from security properties of its components. This is essential for asserting security of complex protocols.

Unfortunately, these properties are not implied by the basic security. In the literature, the strongest and also the most realistic formalization of concurrent security is the notion of Universal Composability (UC) [Can01]: It considers the concurrent execution of an unbounded number of instances of the analyzed protocol, in an arbitrary, and adversarially controlled, network environment. It also supports modular analysis of protocols. But, these strong properties come at a price: Many natural functionalities cannot be realized with UC security in the *plain model,* where players only have access to authenticated communication channels; some additional trusted

set-up is necessary [CF01, CKL03]; furthermore, the need for additional trusted set up extends to any protocol that only guarantees a concurrent extension of basic security [Lin04]. A large body of works (e.g. [CLOS02, BCNP04, KLP05, CPS07, GO07, Kat07, CDPW07]) have shown that indeed, with the appropriate trusted set-ups, UC-security becomes feasible. However, in many situations, trusted set-up is hard to come by (or at least expensive). It is thus important to have a notion of concurrent security that can be achieved in the plain model.

Concurrent Security in the Plain model. *Security with super-polynomial simulators (SPS)* [Pas03a] is a relaxation of UC security that allows the adversary in the ideal execution to run in super-polynomial time. Informally, this corresponds to guaranteeing that "any polytime attack that can be mounted against the protocol can also be mounted in the ideal execution—albeit with super-polynomial resources." Although SPS security is sometimes weaker than basic security, it often provides an adequate level of security. In constrast to basic security, however, SPS directly considers security in the concurrent setting. Protocols that realize practically any functionality with SPS security in the plain model were shown based on sub-exponential hardness assumptions [Pas03a, BS05, LPV09]. Very recently, improved constructions are presented [CLP10, GGJS12, LPV12] that are based on only standard polynomial-time hardness assumptions.

One drawback of SPS security that it is not closed under composition; thus it is not a convenient basis for modular analysis of protocols. Angel-based UC security [PS04] is a framework for notions of security that provides similar security guarantees as SPS and at the same supports modular analysis. Specifically, angel-based security considers a model where both the adversary and the simulator have access to an oracle (an "angel") that allows some judicious use of super-polynomial resources. Since the angels can be implemented in super-polynomial time, for any angel, angel-based security implies SPS security. Furthermore, akin to UC security, angel-based UC security, with any angel, can be used as a basis for modular analysis. Prabhakaran and Sahai [PS04] exhibited an angle with respect to which practically all functionalities can be securely realized; later another angle is given by [MMY06]; both constructions, however, rely on some non-standard hardness assumptions.

Recently, Canetti, Lin and Pass [CLP10] proposed a new notion of security, called *UC with super-polynomial time helpers*. This notion is very similar to the angel-based security where both the adversary and the simulator have access to a helper that provides some super-polynomial time help through a limited interface. Like angel-based security, UC security with super-polynomial time helpers implies SPS security. But, unlike angel-based security where angels are basically non-interactive and stateless, the helpers are *highly interactive and stateful*. Canetti, Lin and Pass [CLP10] then constructed protocols that realize practically all functionalities with respect to a particular super-polynomial-time interactive helper, based on the existence of trapdoor permutations.

Summarizing the state-of-the-art, there are constructions [CLP10, GGJS12, LPV12] of protocols satisfying a meaningful notion of concurrent security—SPS security—in the plain model based on standard polynomial time hardness assumptions. Furthermore, the construction of [CLP10] also supports modular analysis (the constructions of [GGJS12, LPV12] are better in terms of round-complexity—they only require a constant number of communication rounds—but they only acheives "non-composable" SPS security).

However, all these constructions are *non-black-box*, that is, the constructed protocols make non-black-box use of the underlying primitives. In fact, these constructions all follow the "Feige-Shamir" paradigm [FS90]: The protocols contain "trapdoors" embedded into the messages of the protocol, allowing a super-polynomial time simulator to extract the trapdoor and simulate messages in the protocol by "proving that it knows the trapdoor". In general, protocols following this approach seems hard to turn into a "practical" protocol for secure computations; as such, there result should only be viewed as "feasibility results" regarding concurrent secure computation without set-up, but not candidates for practical purposes.

In contrast, black-box constructions that only use the underlying primitives through their in-

put/output interfaces, are often much more efficient and are more suitable for implementation. Therefore, a series of recent works [DI05, IKLP06, IPS08, LP07, Wee10, Goy11] have focused on constructing *black-box construction of secure computation protocols*, as an important step towards bringing secure multi-party computation closer to the practice. However, their constructions are all in either the stand-alone setting or rely on strong trusted set-ups (e.g., trusted hardware). This leaves open the following basic question:

> *Can we obtain a* black-box *construction of concurrently secure protocols in the plain model (preferrably based only standard polynomial-time assumptions)?*
>
> *Can we have such a black-box construction that also satisfies a notion of security supporting composability?*

1.1 Our Results

We present a black-box construction of protocols that satisfy UC security with super-polynomial time helper for a specific helper, based on the existence of a stand-alone semi-honest oblivious transfer (OT) protocols; that is, the weakest possible assumption. The framework of UC with super-polynomial time helper of [CLP10] is formalized through the extended UC (EUC) framework of [CDPW07]; it is identical to the standard UC model [Can00] except that the corrupted parties (and the environement) have access to an additional super-polynomial time entity $\mathcal{H}$, called a helper functionality.

Main Theorem (Informally Stated): *Assume the existence of stand-alone semi-honest oblivious transfer protocols. Then there exists a sub-exponential-time computable interactive machine $\mathcal{H}$ such that for any "well-formed" polynomial-time functionality $\mathcal{F}$, there exists a protocol that realizes $\mathcal{F}$ with $\mathcal{H}$-EUC security, in the plain model. Furthermore, the protocol makes only black-box calls to the underlying oblivious transfer protocol.*

As far as we know, this is the first black-box construction of secure multi-party computation protocols that achieve any non-trivial notion of concurrent security in the plain model (without any trusted-set up, and without assuming an honest majority).

The main technical tool used in our construction is a new notion of a commitment that is secure against adaptive chosen commitment attack (CCA security). The notion of CCA secure commitments was previously introduced in [CLP10]. Roughly speaking, a tag-based commitment scheme (i.e., commitment scheme that take an identifier—called the tag—as an additional input) is said to be *CCA-secure* if the value committed to using the tag id remains hidden even if the receiver has access to a (super-polynomial time) oracle that "breaks" commitments using any tag $\mathsf{id}' \neq \mathsf{id}$, where by breaking, it means the oracle returns a decommitment of the commitment. Thus the oracle is called a decommitment oracle. In [CLP10], a commitment scheme that is CCA-secure w.r.t. a decommiment oracle is constructed based on the minimal assumption of one-way functions. However, their construction is non-black-box. In this work, to obtain black-box secure computation protocols, we need a new *black-box* construction of a CCA-secure commitment scheme. Towards this, we weaken the notion of CCA security w.r.t. decommitment oracle to instead consider an oracle that "breaks" commitments by returning only the unique committed value (and not the decommitment information) of the commitments (or $\perp$ if there is no unique committed value); we call this the committed-value oracle. We then provide a black-box construction of a commitment scheme that is CCA-secure w.r.t. the committed-value oracle.

Theorem (Informally Stated): *Assume the existence of one-way functions. Then, for every $\epsilon > 0$, there exists an $O(n^{\epsilon})$-round commitment scheme that is CCA-secure w.r.t. the committed-value oracle and only relies on black-box access to one-way functions (where n is the security parameter).*

We next show that the notion of CCA-secure commitments intuitively is better behaved than traditional notions of non-malleability [DDN00] in the context of black-box construction of concurrently secure protocol. On a very high-level (and significantly oversimplifying), CCA security of commitment schemes allow us to deal with "cut-and-choose" techniques (typically used in black-box constructions) in concurrent executions, while ensuring hiding of commitments in other executions.

1.2 Outline

In Section 2, we define the notion of CCA-security w.r.t. the committed-value oracle. (Notations and definitions of basic primitives appear in Appendix A.) In Section 4, we present our black-box robust CCA-secure commitment scheme; and provide the proof of security in Appendix B. We recall the notion of UC security with super-polynomial time helper in Appendix C, and show how to achieve this security notion using CCA-secure commitments in a black-box way in Section 3.

2 Definition of CCA-Secure Commitments

We assume familiarity with the definition of commitment schemes and the statistically/computational binding and statistically/computational hiding properties. Unless specified otherwise, by a commitment scheme, we mean one that is statistically binding and computationally hiding. A *tag-based commitment schemes* with $l(n)$-bit identities [PR05, DDN00] is a commitment scheme where, in addition to the security parameter 1^n, the committer and the receiver also receive a "tag"—a.k.a. the identity—id of length $l(n)$ as common input.

2.1 CCA-Security w.r.t. Committed Value Oracle

Let $\langle C, R\rangle$ be a tag-based commitment scheme with $l(n)$-bit identities. A committed-value oracle $\mathcal{O}$ of $\langle C, R\rangle$ acts as follows in interaction with an adversary A: it participates with A in many sessions of the commit phase of $\langle C, R\rangle$ as an honest receiver, using identities of length $l(n)$, chosen adaptively by A. At the end of each session, if the session is *valid*, it reveals the unique committed value of that session to A; otherwise, it sends $\perp$. (If a session has multiple committed values, the decommitment oracle also returns $\perp$. The statistically binding property guarantees that this happens with only negligible probability.) Loosely speaking, a tag-based commitment scheme $\langle C, R\rangle$ is said to be CCA-secure w.r.t. the committed-value oracle, if the hiding property of the commitment holds even with respect to adversaries with access to the committed-value oracle $\mathcal{O}$. More precisely, denote by $A^{\mathcal{O}}$ the adversary A with access to the committed-value oracle $\mathcal{O}$. Let $\mathsf{IND}_b(\langle C, R\rangle, A, n, z)$, where $b \in \{0,1\}$, denote the output of the following probabilistic experiment: on common input 1^n and auxiliary input z, $A^{\mathcal{O}}$ (adaptively) chooses a pair of challenge values $(v_0, v_1) \in \{0,1\}^n$—the values to be committed to—and an identity $\mathsf{id} \in \{0,1\}^{l(n)}$, and receives a commitment to v_b using identity id. Finally, the experiment outputs the output y of $A^{\mathcal{O}}$; the output y is replaced by $\perp$ if during the execution A sends $\mathcal{O}$ any commitment using identity id (that is, any execution where the adversary queries the decommitment oracle on a commitment using the same identity as the commitment it receives, is considered invalid). Let

Definition 1 (CCA-secure Commitments.)**.** *Let $\langle C, R\rangle$ be a tag-based commitment scheme with $l(n)$-bit identities. We say that $\langle C, R\rangle$ is* CCA-secure w.r.t. the committed-value oracle*, if for every $\mathcal{PPT}$ ITM A, the following ensembles are computationally indistinguishable:*

- $\{\mathsf{IND}_0(\langle C, R\rangle, A, n, z)\}_{n\in N, z\in\{0,1\}^*}$
- $\{\mathsf{IND}_1(\langle C, R\rangle, A, n, z)\}_{n\in N, z\in\{0,1\}^*}$

2.1.1 k-Robustness w.r.t. Committed-Value Oracle

Consider a man-in-the-middle adversary that participates in an *arbitrary* left interaction with a *limited number of rounds*, while having access to a committed oracle. Roughly speaking, $\langle C, R\rangle$ is k-robust if the (joint) output of every k-round interaction, with an adversary having access to the oracle $\mathcal{O}$, can be simulated without the oracle. In other words, having access to the oracle does not help the adversary in participating in any k-round protocols much.

Definition 2. *Let $\langle C, R\rangle$ be a tag-based commitment scheme with $l(n)$-bit identities. We say that $\langle C, R\rangle$ is* k-robust w.r.t. the committed-value oracle*, if there exists a simulator S, such that, for every $\mathcal{PPT}$ adversary A, the following two conditions hold.*

Simulation: *For every $\mathcal{PPT}$ k-round ITM B, the following two ensembles are computationally indistinguishable.*

- $\left\{\mathsf{output}_{B,A^{\mathcal{O}}}[\langle B(y), A^{\mathcal{O}}(z)\rangle(1^n, x)]\right\}_{n\in N,x,y,z\in(\{0,1\}^*)^3}$
- $\left\{\mathsf{output}_{B,S^A}[\langle B(y), S^{A(z)}\rangle(1^n, x)]\right\}_{n\in N,x,y,z\in(\{0,1\}^*)^3}$

where $\mathsf{output}_{A,B}[\langle B(y), A(z)\rangle(x)]$ denote the joint output of A and B in an interaction between them, on common input x and private inputs z to A and y to B respectively, with uniformly and independently chosen random inputs to each machine.

Efficiency: *There exists a polynomial t and a negligible function μ, such that, for every $n \in N$, $z \in \{0,1\}^*$ and $x \in \{0,1\}^*$, and every polynomial T, the probability that S with oracle access to $A(z)$ and on input $1^n, x$, runs for more than $T(n)$ steps is smaller than $\frac{t(n)}{T(n)} + \mu(n)$.*

The following proposition shows that to construct a robust CCA-secure commitment scheme for identities of length n, it suffices to construct one for identities of length $\ell(n) = n^\varepsilon$. The same proposition is established in [CLP10] for robust CCA-security w.r.t. decommitment oracles, and the proof there also applies to CCA-security w.r.t. committed-value oracles; we omit the proof here.

Proposition 1. *Let ε be any constant such that $0 < \varepsilon < 1$, ℓ a polynomial such that $\ell(n) = n^\varepsilon$, and $\langle C, R\rangle$ a k-round robust CCA-secure commitment scheme (w.r.t. the committed-value oracle) with ℓ-bit identities. Then assuming the existence of one-way functions, there exists a robust $k+1$-round CCA-secure commitment scheme $\langle \hat{C}, \hat{R}\rangle$ (w.r.t. the committed-value oracle) with n-bit identities.*

3 Black-Box UC-Secure Protocols with Super-Polynomial Helpers

We consider the model of UC with super-polynomial helper introduced in [CLP10]. At a very high-level, this model is essentially the same as the UC-model introduced by [Can00], except that both the adversary and the environment in the real and ideal worlds have access to a super-polynomial time functionality that acts as a helper. A formal definition of the model is presented in Appendix C. In this section, we show the following theorem:

Theorem 1. *Let δ be any positive constant. Assume the existence of a T'_{OT}-round stand-alone semi-honest oblivious transfer protocol. Then there exists a super-polynomial time helper functionality $\mathcal{H}$, such that, for every well-formed functionality*[1] *$\mathcal{F}$, there exists a $O(\max(n^\delta, T'_{OT}))$-round protocol Π that $\mathcal{H}$-EUC-emulates $\mathcal{F}$. Furthermore, the protocol Π only uses the underlying oblivious transfer protocol in a black-box way.*

[1] See [CLOS02] for a definition of well-formed functionalities.

Towards this theorem, we need to first exhibit a super-polynomial time helper functionality $\mathcal{H}$. Roughly speaking, $\mathcal{H}$ simply acts as the committed-value oracle of a CCA secure commitment scheme. More precisely, consider the following two building blocks: First, given any $T'_{OT}(n)$-round stand-alone semi-honest OT protocol, it follows from previous works [IKLP06, Hai08] that there exists an $T_{OT}(n)$-round OT protocol $\langle S, R\rangle$ that is secure against a malicious sender and a semi-honest receiver—called mS-OT protocol for short—that only relies on black-box access to the semi-honest OT protocol; furthermore $T_{OT} = O(T'_{OT}(n))$. Second, we need a $T_{OT}(n)$-robust CCA-secure commitment scheme $\langle C, R\rangle$, whose committed-value oracle $\mathcal{O}$ can be computed in sub-exponential time.[2] As we will show in the next section (see Theorem 2), such a protocol exists with $O(\max(T_{OT}, n^\delta)) = O(\max(T'_{OT}, n^\delta))$ rounds, relying on the underlying OWF in a black-box way. Since OWFs can be constructed from a semi-honest OT protocol in a black-box way. Therefore, we have that the second building block can also be based on the semi-honest OT protocols in a black-box way.

Consider a helper functionality $\mathcal{H}$ that "breaks" commitments of $\langle C, R\rangle$ in the same way as its committed-value oracle $\mathcal{O}$ does, subject to the condition that player P_i in a protocol instance *sid* can only query the functionality on commitments that uses identity (P_i, sid). More precisely, every party P_i in a secure computation can simultaneously engage with $\mathcal{H}$ in multiple sessions of the commit phase of $\langle C, R\rangle$ as a committer using identity P_i, where the functionality simply forwards all the messages internally to the committed-value oracle $\mathcal{O}$, and forwards P_i the committed value returned from $\mathcal{O}$ at the end of each session. Since the committed-value oracle $\mathcal{O}$ can be computed in sub-exponential time, this functionality can also be implemented in sub-exponential time. A formal description of the functionality appears in Figure 6 in Appendix D.

We show that Theorem 1 holds w.r.t. the helper functionality defined above in two steps. First, note that to realize any well-formed functionality in a black-box way, it suffices to realize the *ideal oblivious transfer functionality* $\mathcal{F}_{\mathsf{OT}}$ [Rab05, EGL85]. This is because it follows from previous works [Kil92, BOGW88, GMW91, IPS08] that every functionality can be UC securely implemented in the $\mathcal{F}_{OT}$-hybrid model, even w.r.t. super-polynomial time environments. Based on previous works, [CLP10] further shows that by considering only dummy adversaries and treating environments with access to a super-polynomial functionality $\mathcal{H}$ as sub-exponential time machines, we have that every functionality can be $\mathcal{H}$-EUC securely implemented in the $\mathcal{F}_{\mathsf{OT}}$ model. Formally, we have the following lemma from [CLP10].

Lemma 1. *Fix any super-polynomial time functionality $\mathcal{H}$. For every well-formed functionality $\mathcal{F}$, there exists a constant-round $\mathcal{F}_{OT}$-hybrid protocol that $\mathcal{H}$-EUC-emulates $\mathcal{F}$.*

Next we show how to implement the $\mathcal{F}_{OT}$ functionality in the $\mathcal{H}$-EUC model. Then combining with Lemma 1, we conclude Theorem 1.

Lemma 2. *Let δ be any positive constant. Assume the existence of a T'_{OT}-round semi-honest oblivious transfer protocol. Then there exists a $O(\max(n^\delta, T'_{OT}))$-round protocol Π_{OT} that $\mathcal{H}$-EUC-emulates $\mathcal{F}_{\mathsf{OT}}$. Furthermore, the protocol Π_{OT} only uses the underlying oblivious transfer protocol in a black-box way.*

3.1 Overview of the OT Protocol Π_{OT}

In this section we first provide an overview of our black-box construction of $\mathcal{H}$-EUC secure OT protocol Π_{OT}, in comparison with the black-box construction of an OT protocol secure against both malicious players from a mS-OT protocol of [IKLP06, Hai08]. Roughly speaking, the protocol

[2] This can be instantiated by simply using a normal T_{OT}-robust CCA secure commitments with an exponential time committed value $\mathcal{O}$, with a "scaled-down" security parameter.

of [IKLP06, Hai08], relying on a stand-alone mS-OT protocol $\langle S, R\rangle$, proceeds in the following four stages:

Stage 1 (Receiver's Random Tape Generation) The sender and the receiver jointly decide the receiver's inputs and random tapes in Stage 2 using $2n$ parallel "coin tossing in the well" executions.

Stage 2 (OT with Random Inputs) The sender and the receiver perform $2n$ parallel OT executions of $\langle S, R\rangle$ using *random* inputs (s_j^0, s_j^1) and r_j respectively, where the receiver's inputs r_j's (and its random tapes) are decided in Stage 1.

Stage 3 (Cut-and-Choose) A random subset $Q \subset [2n]$ of n locations is chosen using a 3-round coin-tossing protocol where the sender commits to a random value first. (Thus the receiver knowing that random value can bias the coin-tossing output.) The receiver is then required to reveal its randomness in Stage 1 and 2 at these locations, which allows the sender to check whether the receiver behaved honestly in the corresponding OT executions. The randomness of the receiver at the rest of locations remains hidden.

Stage 4 (OT Combiner) Finally, for these locations $j \notin Q$ that are not open, the receiver sends $\alpha_j = u \oplus c_j$ where u is the receiver's true input. The sender replies with $\beta_0 = v_0 \oplus (\bigoplus_{j \notin Q} s_j^{\alpha_j})$ and $\beta_1 = v_1 \oplus (\bigoplus_{j \notin Q} s_j^{1-\alpha_j})$. The honest receiver obtains $s_j^{c_j}$'s through the OT execution, and thus can always recover v_u.

At a very high-level, the protocol of [IKLP06, Hai08] augments security of the mS-OT protocol $\langle S, R\rangle$ to handle malicious receivers, by adding the cut-and-choose (as well as the random tape generation) stage to enforce the adversary behaving honestly in *most* (Stage 2) OT executions. (This is in a similar spirit as the non-black-box approach of requiring the receiver to prove that it has behaved honestly.) Then the security against malicious receivers can be based on that against semi-honest receivers of $\langle S, R\rangle$.

Wee [Wee10] further augmented the stand-alone security of the protocol of [IKLP06, Hai08] to achieve parallel security, that is, obtaining a protocol that is secure against man-in-the-middle adversaries that simultaneously acts as sender and receiver in many parallel executions. Towards this, Wee instantiates the commitments in the coin-tossing sub-protocols of the protocol of [IKLP06, Hai08], with ones that are satisfy a notion of "non-malleable w.r.t. extraction". Roughly speaking, non-malleability w.r.t. extraction [Wee10] is a weaker notion than non-malleability of [DDN00, LPV08]; it guarantees that no matter what values the adversary is receiving commitments to, the committed values extracted out of the commitments from the adversary (with over-extraction) are indistinguishable. This guarantees that a simulator can bias the coin-tossing output by extracting the committed values from the adversary while the adversary cannot, as otherwise, by non-malleability w.r.t. extraction, it could do so even if the honest player sends a commitment to 0 instead of its true random challenge q. However, this is impossible as in this case no information of q is revealed. In other words, the coin-tossing protocol when instantiated with a non-malleable w.r.t. extraction commitment becomes parallel secure, relying which Wee shows that the OT protocol also becomes parallel secure.

Towards $\mathcal{H}$-EUC-Secure OT protocols, we need to further overcome two problems.

First, we need to go from parallel security to concurrent security. In other words, we need coin-tossing protocols that are concurrently secure. Informally speaking, non-malleability w.r.t. extraction guarantees that the simulator can extract the committed values of commitments from the adversary (to bias the output of the coin-tossing) while keeping the commitment to the adversary hiding amid rewindings (to ensure that the adversary cannot bias the output). However, this only holds in the parallel setting, as non-malleability only guarantees hiding of a commitment when

values of the commitments from the adversary are extracted in parallel at the end of the execution. But, in the concurrent setting, the simulator needs to extract the committed values from the adversary in an on-line manner, that is, whenever the adversary successfully completes a commitment the committed value is extracted. To resolve this problem, we resort to CCA-secure commitments, which guarantees hiding of a commitment even when the committed values are extracted (via the committed-value oracle) concurrently and immediately after each commitment. Now, instantiating the commitment scheme in the coin-tossing protocols with a CCA-secure commitment yields a coin-tossing protocol that is concurrently secure.

The second problem is that to achieve $\mathcal{H}$-EUC-security (similar to UC-security), we need to design a protocol that admits straight-line simulation. The simulator of a OT protocol has three tasks: It needs to simulate the messages of the honest sender and receiver, extract a choice from the adversary when it is acting as a receiver, and extract two inputs when it is acting as a sender. To achieve the first two tasks, the original simulation strategy in [IKLP06, Hai08, Wee10] uses rewindings to breaking the non-malleable commitments from the adversary to bias coin-tossing. When using CCA-secure commitments, the simulator can extract the committed values in a straight-line, by forwarding the commitment from the adversary to the helper functionality $\mathcal{H}$ that breaks the CCA commitments using brute force. For the last task, the original simulation strategy uses the simulator of the mS-OT protocol $\langle S, R\rangle$ against malicious senders to extract the adversary's inputs s_j^b's in the Stage 3 OT executions, which then allows extraction of the real inputs v_0 and v_1 from the last message. However, the simulator of the mS-OT protocol may use rewindings. To solve this, one way is to simply assume a mS-OT protocol that has a straight-line simulator. We here however, present a different solution.

In our protocol, the sender and the receiver participate in parallel "coin tossing in the well" executions to decide the sender's random inputs s_j^b (and random tapes) in the Stage 3 OT executions (besides the receiver's inputs and random tapes). Since the simulator can bias the coin-tossing in a straight line, it can determine the sender's inputs s_j^b's, which allows extraction of the sender's true inputs. For this to work, we need to make sure that a malicious sender would indeed uses the outputs of coin-tossing as inputs in the OT executions. Towards this, we again use the cut-and-choose technique: After the OT execution, the sender is required to reveal its randomness in the coin-tossing and OT execution at a randomly chosen subset of locations. The cut-and-choose technique guarantees that a malicious sender will behave consistently in most OT executions. Therefore the simulator extracts (through the coin-tossing executions) the inputs s_j^b's correctly at *most* locations. However, in the protocol of [IKLP06, Hai08, Wee10], to recover the real inputs v_0 and v_1, the simulator needs to obtain *all* s_j^b's correctly. To bridge the gap, we modify the protocol to have the sender compute a random secret-sharing $\left\{a_j^b\right\}$ of each input v_b and hide each share using the appropriate s_j^b, that is, it sends $a_j^b \oplus s_j^{b\oplus\alpha}$ for every j (that is not open in the cut-and-choose procedures). Then, the simulator, able to extract most s_j^b's correctly, can recover enough shares to decode to the real inputs correctly. In contrast, a malicious receiver that is enforced to behave honestly in most OT executions by the cut-and-choose procedure, cannot obtain enough shares for both inputs and thus can only recover one of them. Finally, we remark that as in [Wee10], to avoid over-extraction from the secret shares, we use the technique used in [CDSMW08, CDSMW09], which adds a another cut-and-choose procedure. A formal description of our OT protocol Π_{OT} that implements $\mathcal{F}_{\mathsf{OT}}$ is provided in Figure 1; a formal proof of security of Π_{OT} (i.e., Lemma 2) appears in Appendix D.1.

OT protocol Π_{OT}

Inputs: The sender and receiver receive common input a security parameter 1^n and private inputs (v_0, v_1) and $u \in \{0,1\}$ respectively.

Stage 1: The sender chooses a random subset $\Gamma_R \subseteq [20n]$ of size n and commits to Γ_R using $\langle C, R\rangle$.

The receiver chooses a random subset $\Gamma_S \subseteq [20n]$ of size n and another random subset $\Gamma \subseteq [18n]$ of size n; it then commits to both Γ_S and Γ using $\langle C, R\rangle$.

Stage 2 (Coin-Tossing):

Receiver Random-Tape Generation: The receiver chooses $20n$ random strings $(a_1^R, \ldots a_{20n}^R)$ and commits to them using $\langle C, R\rangle$. The sender sends $20n$ random strings $(b_1^R, \ldots b_{20n}^R)$. The receiver calculates $r_i^R = a_i^R \oplus b_i^R$ for every $i \in [20n]$, and interprets r_i^R as $c_i \| \tau_i^R$, where c_i will be used as the receiver's input bit, and τ_i^R the random tape in the OT executions below.

Sender Random-Tape Generation: The sender chooses $20n$ random strings $(a_1^S, \ldots a_{20n}^S)$ and commits to them using $\langle C, R\rangle$. The receiver sends $20n$ random strings $(b_1^S, \ldots b_{20n}^S)$. The sender calculates $r_i^S = a_i^S \oplus b_i^S$ for every $i \in [20n]$, and interprets r_i^S as $s_i^0 \| s_i^1 \| \tau_i^S$, where s_i^0 and s_i^1 will be used as the sender's two input bits, and τ_i^S the random tape in the OT executions below.

Stage 3 (OT with Random Inputs): The sender and the receiver participates in $20n$ executions of the OT protocol $\langle S, R\rangle$ in parallel, where the sender acts as S and the receiver acts as R. In the i^{th} execution of $\langle S, R\rangle$, the sender uses inputs s_i^0, s_i^1 and random tape r_i^S and the receiver uses input c_i and random tape r_i^R. At the end of the execution, the receiver obtains outputs $\tilde{s}_1 \ldots \tilde{s}_{20n}$.

Stage 4 (Cut-and-Choose—Honesty Checking):

Sender Honesty Checking: The receiver opens Γ_S and sender responds as follows: for every $j \in \Gamma_S$, the sender opens the j^{th} commitments of $\langle C, R\rangle$ in Stage 2 to $\tilde{a}_j^S$. The receiver checks if the openings are valid, and if for every $j \in \Gamma_S$, the sender acted honestly in the j^{th} OT execution according to $\tilde{a}_j^S \oplus b_j^S$. The receiver aborts if not.

Receiver Honesty Checking: The sender opens Γ_R and receiver responds as follows: for every $j \in \Gamma_R$, the receiver opens the j^{th} commitments of $\langle C, R\rangle$ in Stage 2 to $\tilde{a}_j^R$. The sender checks if the openings are valid and if for every $j \in \Gamma_R$, the receiver acted honestly in the j^{th} OT execution according to $\tilde{a}_j^R \oplus b_j^R$. The sender aborts if not.

Stage 5 (Combiner): Set $\Delta = [20n] - \Gamma_R - \Gamma_S$ (i.e., Δ is the set of unopened locations). For every $i \in \Delta$ The receiver computes $\alpha_i = u \oplus c_i$ and sends α_i. The sender responds as follows: It computes a $10n$-out-of-$18n$ secret-sharing of v_0; without loss of generality, we index shares in that secret-sharing with elements in Δ; let the secret-sharing be $\rho^0 = \left\{\rho_i^0\right\}_{i\in\Delta}$. Similarly, it also computes a $10n$-out-of-$18n$ secret-sharing $\rho^1 = \left\{\rho_i^1\right\}_{i\in\Delta}$ for v_1. Then the sender computes $\beta_i^b = \rho_i^b \oplus s_i^{b\oplus\alpha_i}$ for every $i \in \Delta$ and sends back all the β_i^b's.

The receiver after receiving all the β_i^b's, computes shares corresponding to the u^{th} input as $\tilde{\rho}_i = \beta_i^u \oplus \tilde{s}_i$ for every $i \in \Delta$, and sets $\tilde{\rho} = \{\tilde{\rho}_i\}_{i\in\Delta}$.

Stage 6 (Cut-and-Choose—Consistency Checking): The receiver opens to Γ. Then for every $j \in \Gamma \cap \Delta$, the sender reveals the two inputs $\hat{s}_j^0$ and $\hat{s}_j^1$ and random tape $\hat{\tau}_j^S$ that it uses in the j^{th} OT execution in Stage 3. The receiver checks if the sender acts honestly according to input $(\hat{s}_j^0, \hat{s}_j^1)$ and random tape $\hat{\tau}_j^S$ and aborts if not.

Finally the receiver checks whether $\tilde{\rho}$ computed in Stage 5 is $17n$-close to a valid codeword w (that is, it agrees with w at $17n$ locations), and if for every $j \in \Gamma \cap \Delta$, w_j is equal to $\beta_j^u \oplus \hat{s}_j^{u\oplus\alpha_j}$. If so it outputs the value v encoded in w; otherwise, it aborts.

Figure 1: Our OT protocol Π_{OT} that implements $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{H}$-EUC model

4 Black-Box Robust CCA-Secure Commitments

In this section, we present a *black-box* construction of a robust CCA-secure commitment scheme w.r.t. committed-value oracle based on one-way functions. For simplicity of exposition, the presentation below relies on a non-interactive statistically binding commitment scheme com; this can be replaced with a standard 2-round statistically binding commitment scheme using standard techniques[3]. Our construction makes use of previous black-box constructions of extractable commitments and trapdoor commitment scheme. So let's start by reviewing them.

Extractable Commitments Intuitively, an extractable commitment is one such that for any machine C^* sending a commitment, a committed value can be extracted from C^* if the commitment it sends is valid; otherwise, if the commitment is invalid, then no guarantee is provided, that is, an arbitrary garbage value may be extracted. This is known as the "over-extraction" problem. (See Appendix A.7 for a formal definition.) As shown in [PW09], the following protocol used in the works of [DDN00, PRS02, Ros04] (also [Kil88]) yields a black-box extractable commitment scheme ExtCom: To commit to a value $v \in \{0,1\}^m$, the committer and receiver on common input a security parameter 1^n, proceed as follows:

Commit: The committer finds n pairs of random shares $\left\{v_0^i, v_1^i\right\}_{i\in[n]}$ that sum up to v, (i.e., $v_0^i \oplus v_1^i = v$ for all $i \in [n]$) and commits to them in parallel using the non-interactive statistically binding commitment scheme com. Let c_b^i be the commitment to v_b^i.

Challenge: The receiver sends a n-bit string $ch \in \{0,1\}^n$ sampled at random.

Reply: The committer opens commitments $c_{ch_i}^i$ for every $i \in [n]$.

To decommit, the sender sends v and opens the commitments to all n pairs of strings. The receiver checks whether all the openings are valid and also $v = v_0^i \oplus v_1^i$ for all i.

It is proved in [PW09] that ExtCom is extractable. Furthermore, the commitment scheme has the property that from any two accepting transcripts of the commit stage that has the same commit message but different challenge messages, the committed value can be extracted. This property is similar to the notion of special-soundness for interactive proof/argument systems; here we overload this notion, and refer to this special extractability property of ExtCom as special-soundness.

In our construction, we will actually need an extractable commitment scheme to a string $\sigma \in \{0,1\}^m$ for which we can open any subset of the bits in σ without compromising the security (i.e. hiding) of the remaining bits. As shown in [PW09], we may obtain such a scheme $\mathsf{PExtCom}$ by running ExtCom to commit to each bit of σ in parallel. It is easy to see that $\mathsf{PExtCom}$ is also special-sound in the sense that, given two accepting transcripts of $\mathsf{PExtCom}$ that have the same commit message and two challenge messages that contain a pair of different challenges for every ExtCom commitment, the committed string σ can be extracted. We call such two transcripts a pair of $\mathsf{admissible\ transcripts}$ for $\mathsf{PExtCom}$.

Trapdoor Commitments Roughly speaking, a trapdoor commitment scheme is a computationally biding and computationally hiding commitment scheme, such that, there exists a simulator that can generate a simulated commitment, and later open it to any value. (See Appendix A.8 for a formal definition.) Pass and Wee [PW09] presented a black-box trapdoor bit commitment scheme $\mathsf{TrapCom}$. To commit to a bit σ, the committer and the receiver on common input 1^n do:

Stage 1: The receiver picks a random string challenge $e = (e_1, \ldots, e_n)$ and commits to e using the non-interactive statistically binding commitment scheme com.

[3]This can be done by sending a first message of a 2-round commitment scheme at the beginning of the protocol, and using the second message of the 2-round commitment scheme w.r.t. that first message as a non-interactive commitment in the rest of the protocol.

Stage 2: The committer prepares $v_1, \ldots, v_n$. Each v_i is a 2×2 0,1-matrix given by

$$\begin{pmatrix} v_i^{00} & v_i^{01} \\ v_i^{10} & v_i^{11} \end{pmatrix} = \begin{pmatrix} \eta_i & \eta_i \\ \sigma \oplus \eta_i & \sigma \oplus \eta_i \end{pmatrix}$$

where η_i is a random bit. The sender commits to $v_1, \ldots, v_n$ using $\mathsf{PExtCom}$ (i.e., committing using ExtCom bit by bit in parallel). In addition, the sender prepares $(a_1^0, a_1^1), \ldots, (a_n^0, a_n^1)$ where a_i^β is the opening to $v_i^{\beta 0}, v_i^{\beta 1}$ (i.e., either the top or bottom row of v_i).

Stage 3: The receiver opens to the challenge $e = (e_1, \ldots, e_n)$; the sender responds with $a_1^{e_1}, \ldots, a_n^{e_n}$.

To decommit, the sender sends σ. In addition, it chooses a random $\gamma \in \{0, 1\}$, sends γ, sends the openings to values $v_i^{0\gamma}, v_i^{1\gamma}$ for $i = 1, 2, \ldots, n$ (i.e., either the left columns or the right columns of all the matrices). The receiver checks that all the openings are valid, and also that $\sigma = v_1^{0\gamma} \oplus v_1^{1\gamma} = \cdots = v_n^{0\gamma} \oplus v_n^{1\gamma}$.

As shown in [PW09], the protocol $\mathsf{TrapCom}$ is trapdoor, following a Goldreich-Kahan [GK96] style proof; moreover, by running $\mathsf{TrapCom}$ in parallel, we obtain a trapdoor commitment scheme $\mathsf{PTrapCom}$ for multiple bits. Furthermore, since Stage 2 of the protocol $\mathsf{TrapCom}$ is simply an execution of $\mathsf{PExtCom}$, given any two *admissible transcripts* of Stage 2, the matrices $v_1, \ldots, v_n$ prepared in Stage 2 can be extracted; we show that from these matrices, the actual bit committed in the $\mathsf{TrapCom}$ commitment can be extracted, provided that the commitment is valid and has a unique committed value. We call this, again, the special-soundness of $\mathsf{TrapCom}$. It is easy to see that the notion of special soundness (and admissible transcripts) can be easily extended for $\mathsf{PTrapCom}$. The formal definition and proof of special-soundness of $\mathsf{TrapCom}$ (and $\mathsf{PTrapCom}$) appear in Appendix B.1.

4.1 Overview of Our Construction

The CLP Construction: At a very high level, the CLP construction proceeds by having the committer first commit to the value v using a normal statistically binding commitment com, followed by a sequence of $poly(n)$ $\mathcal{WISSP}$ proofs of the committed value. The $\mathcal{WISSP}$ proofs are the non-black-box component of the CLP construction, but are crucial for achieving CCA-security. Recall that proving CCA-security w.r.t. $\mathcal{O}$ amounts to showing that the views of A in experiments IND_0 and IND_1 are indistinguishable (when A has oracle access to $\mathcal{O}$). Let us refer to the adversary's interaction with C as the *left interaction*, and its interactions with $\mathcal{O}$ as the *right interactions*. The main hurdle in showing the indistinguishability of IND_0 and IND_1 is that the oracle $\mathcal{O}$ is not efficiently computable; if it were, indistinguishability would directly follow from the hiding property of the left interaction. The main idea of the security proof of [CLP10] is then to implement the oracle $\mathcal{O}$ by extracting the committed values from the adversary, via "rewinding" the special-sound proofs in the right interactions. The two main technical challenges arises in simulating oracle $\mathcal{O}$.

First, once the simulation starts rewinding the right interactions, A might send new messages also in the left interaction. So, if done naively, this would rewind the left interaction, which could violate its hiding property. To solve this problem, the CLP protocol schedules messages in the special-sound proofs using a special message scheduling (according to the identity of the commitment), called the CLP scheduling, which is a variant of the message scheduling technique of [DDN00, LPV08]. The special message scheduling ensures that for every accepting right interaction with an identity that is different from the left interaction, there exists many points—called $\mathsf{safe\text{-}points}$—in the interaction, from which one can rewind the right interaction without requesting any *new* message in the left interaction.

Second, in the experiment IND_b, the adversary A expects to receive the committed value at the very moment it completes a commitment to its oracle. If the adversary "nests" its oracle

calls, these rewindings become recursive and the running-time of the extraction quickly becomes exponential. To avoid the extraction time from exploding, the simulation strategy in CLP rewinds from safe-points using a concurrent extraction strategy that is similar to that used in the context of concurrent $\mathcal{ZK}$ by Richardson and Killian [RK99].

New Approach: To obtain a black-box construction, our main goal is to replace the $\mathcal{WISP}$ proofs with an "equivalent" black-box component. The key property that the CLP proof relies on is that the protocol contains many *3-round* constructs satisfying that rewinding the last two messages reveals the committed value, but rewinding three messages reveals nothing. It seems that the 3-round commitment scheme PExtCom is a good replacement of $\mathcal{WISP}$ proofs as one such 3-round construct: The special-soundness property of PExtCom ensures that rewinding the last two messages reveals the committed value, and the hiding property ensures that rewinding three messages reveals nothings. It is thus tempting to consider a commitment scheme in which the committer commits to value v using $\text{poly}(n)$ invocations of PExtCom, arranged according to the CLP scheduling; the CLP extraction strategy guarantees that for every accepting right interaction, (the last two messages of) one PExtCom commitment is rewound and a committed value is extracted. Indeed, if a commitment of this scheme is valid, meaning that all the PExtCom commitments contained in it are valid commitments to the same value, the CLP extraction strategy returns the unique committed value. However, if the commitment is invalid, there arises the over-extraction problem: The CLP extraction strategy may extract a garbage value from an invalid PExtCom commitment or from a valid commitment that is inconsistent with the other commitments.

To solve the over-extraction problem, we use the cut-and-choose technique to enforce the committer to give valid and consistent PExtCom commitments. Instead of having the committer commit to v directly, let it commit to a $(n+1)$-*out-of*-$10n$ *Shamir's secret sharing* $s_1, \ldots, s_{10n}$ *of* v using many PExtCom invocations, still arranged according to the CLP scheduling; we refer to all the commitments to the j^{th} share s_j the j^{th} column. After all the PExtCom commitments, the receiver requests the committer to open all the commitments in n randomly chosen columns; the receiver accepts only if each column contains valid commitments to the same value. It follows from the cut-and-choose technique that except with negligible probability, at most n columns may contain invalid or inconsistent commitments. Therefore, when applying the CLP extraction strategy on a commitment of this scheme, it guarantees to extract out a secret-sharing that is .9-close to all the secret-sharing committed to in this commitment. Then by relying on the error-correcting property of the secret sharing, a valid committed value can be reconstructed. The formal analysis is actually more subtle; to avoid over-extraction, we employ the technique used in [CDSMW08, CDSMW09, Wee10], which involves setting the validity condition of the commitment scheme carefully so that invalid commitment can be identified. We refer the reader to Appendix B.3 for more details.

Unfortunately, our use of the cut-and-choose technique brings another problem: The above commitment scheme may not be hiding. This is because, in the last stage, the receiver may request the committer to open an adaptively chosen subset of commitments of PExtCom, the remaining unopened commitments may not be hiding, unless PExtCom were secure against selective opening attack. To resolve this problem, we use the trapdoor commitment scheme PTrapCom to replace PExtCom. Since PTrapCom is trapdoor, it is secure against selective opening attack, and thus the hiding property holds. Furthermore, since Stage 2 of PTrapCom is simply a commitment of PExtCom, we can use Stage 2 of PTrapCom as an implementation of the 3-round construct needed for the CLP scheduling and extraction strategy. More precisely, the commitment scheme proceeds as follow: The committer commits to a $(n+1)$-out-of-$10n$ secret sharing of the value v using many invocations of PTrapCom, where all the invocations share the same Stage 1 message sent at the beginning, followed by all the 3-round Stage 2 executions arranged according to the CLP scheduling, and then all the Stage 3 executions performed in parallel; finally, the committer and the receiver conducts a cut-and-choose consistency check as described above.

It seems that the security proof of our CCA-secure commitment should follow from that of the non-black-box construction of [CLP10]. Unfortunately, due to the fact that the "rewinding slots" of our protocol, that is the commitment of ExtCom, may have over-extraction, whereas the $\mathcal{WISSP}$ proofs in the CLP protocol never has this problem, the technical proof of [CLP10] does not go through; and in Appendix B we rely on a different analysis to show the security of our protocol. A formal description of our CCA secure protocol $\langle C, R\rangle$ in Figure 2.

The robust CCA-secure protocol $\langle C, R\rangle$

Let κ be an arbitrary polynomial, ℓ, η two polynomials such that $\ell(n) = n^\nu$ and $\eta(n) = n^\varepsilon$ for $\nu, \varepsilon > 0$, and L a polynomial such that $L(n) = \max(\kappa(n)+\eta(n), 4\ell(n)\eta(n))$. To commit to a value v, the committer C and the receiver R, on common input 1^n and the identity $\mathsf{id} \in \{0,1\}^{\ell(n)}$ of the committer C do:

Stage 1: The receiver sends the Stage 1 message of a commitment of $\mathsf{PTrapCom}$. That is, a commitment of com to a randomly chosen string challenge $e = (e_1, \ldots, e_n)$.

Stage 2: The committer C prepares a $(n+1)$-out-of-$10n$ Shamir's secret sharing $s_1, \ldots, s_{10n}$ of the value v, and commits to these shares using Stage 2 of the protocol $\mathsf{PTrapCom}$ in parallel, *for $L(n)$ times*; we call the i^{th} parallel commitment the i^{th} *row*, and all the commitments to the i^{th} share s_i the i^{th} *column*.

Messages in the first $4\ell(n)\eta(n)$ rows are scheduled based on the identity id and relies on scheduling pairs of rows according to schedules design_0 and design_1 depicted in Figure 3. More precisely, Stage 2 consist of $\ell(n)$ phases. In phase i, the committer provides $\eta(n)$ sequential $\mathsf{design}_{\mathsf{id}_i}$ pairs of rows, followed by $\eta(n)$ sequential $\mathsf{design}_{1-\mathsf{id}_i}$ pairs of rows. Messages in the rest of the rows are simply arranged sequentially.

Stage 3: The receiver opens the Stage 1 commitment to the challenge e. The committer completes the $10nL(n)$ executions of $\mathsf{PTrapCom}$ w.r.t. challenge e in parallel.

Stage 4 (cut-and-choose): The receiver sends a randomly chosen subset $\Gamma \in [10n]$ of size n. For every $j \in \Gamma$, the committer opens all the commitments in the j^{th} column of Stage 3. The receiver checks that all the openings are valid, and reveal the same committed values s_j.

Decommitment Message: To decommit, the committer sends v, and opens all the commitments in the first row of Stage 2 to $s_1, \ldots, s_{10n}$. The receiver checks all the openings to $s_1, \ldots, s_{10n}$ are valid; furthermore, it checks that $s_1, \ldots, s_{10n}$ is 0.9-close to a valid codeword $w = (w_1, \cdots, w_{10n})$, and for every $j \in \Gamma$, w_j equals to the share s_j revealed in Stage 4.

In other words, a commitment of $\langle C, R\rangle$ is valid if and only if the first row in Stage 2 of the commitment contains valid commitments to shares $s_1, \ldots, s_{10n}$, such that, $s_1, \ldots, s_{10n}$ is 0.9 close to a valid codeword w, and w agrees with all the shares revealed in Stage 4 (i.e., for every $j \in \Gamma$, $w_j = s_j$).

Figure 2: The formal description of the $\kappa(n)$-robust CCA-secure protocol $\langle C, R\rangle$

References

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

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

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

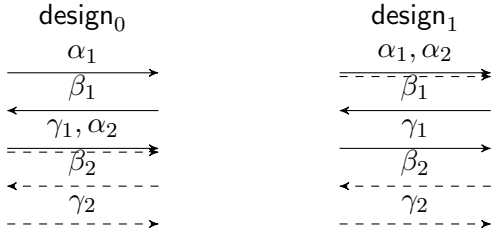

Figure 3: Description of the schedules used in Stage 2 of the protocol. $(\alpha_1, \beta_1, \gamma_1)$ and $(\alpha_2, \beta_2, \gamma_2)$ are respectively the transcripts of a pair of rows in Stage 2 of the protocol $\langle C, R \rangle$.

[BS05] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552, 2005.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, pages 143–202, 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, pages 136–145, 2001.

[Can04] Ran Canetti. Universally composable signature, certification, and authentication. In *CSFW*, pages 219–, 2004.

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In *TCC*, pages 61–85, 2007.

[CDSMW08] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In *TCC*, pages 427–444, 2008.

[CDSMW09] Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Simple, black-box constructions of adaptively secure protocols. In *TCC*, pages 387–402, 2009.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO*, pages 19–40, 2001.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT*, pages 68–86, 2003.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *FOCS*, pages 541–550, 2010.

[CPS07] Ran Canetti, Rafael Pass, and Abhi Shelat. Cryptography from sunspots: How to use an imperfect reference string. In *FOCS*, pages 249–259, 2007.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, pages 378–394, 2005.

[DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. To appear in EUROCRYPT 2012, 2012.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

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

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, 1991.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *CRYPTO*, pages 323–341, 2007.

[Gol01] Oded Goldreich. *Foundations of Cryptography — Basic Tools*. Cambridge University Press, 2001.

[Goy11] Vipul Goyal. Constant round non-malleable protocols using one way functions. In *STOC*, pages 695–704, 2011.

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *TCC*, pages 412–426, 2008.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.

[IKLP06] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. Black-box constructions for secure computation. In *STOC*, pages 99–108, 2006.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, pages 115–128, 2007.

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992.

[KLP05] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent general composition of secure protocols in the timing model. In *STOC*, pages 644–653, 2005.

[Lin04] Yehuda Lindell. Lower bounds for concurrent self composition. In *TCC*, pages 203–222, 2004.

[LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *EUROCRYPT*, pages 52–78, 2007.

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In *TCC*, pages 571–588, 2008.

[LPV09] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. A unified framework for concurrent security: universal composability from stand-alone non-malleability. In *STOC*, pages 179–188, 2009.

[LPV12] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Uc from semi-honest ot. Manuscript, 2012.

[MMY06] Tal Malkin, Ryan Moriarty, and Nikolai Yakovenko. Generalized environmental security from number theoretic assumptions. In *TCC*, pages 343–359, 2006.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4:151–158, 1991.

[Pas03a] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.

[Pas03b] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.

[PR05] Rafael Pass and Alon Rosen. Concurrent non-malleable commitments. In *FOCS*, pages 563–572, 2005.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251, 2004.

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.

[Rab05] Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *Eurocrypt*, pages 415–432, 1999.

[Ros04] Alon Rosen. A note on constant-round zero-knowledge proofs for np. In *TCC*, pages 191–202, 2004.

[Wee10] Hoeteck Wee. Black-box, round-efficient secure computation via non-malleability amplification. To appear in FOCS 2010, 2010.

[Xia11] David Xiao. (nearly) round-optimal black-box constructions of commitments secure against selective opening attacks. In *TCC*, pages 541–558, 2011.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

A General Definitions

A.1 Witness Relations

We recall the definition of a witness relation for a $\mathcal{NP}$ language [Gol01].

Definition 3 (Witness relation)**.** *A* witness relation *for a language $L \in \mathcal{NP}$ is a binary relation R_L that is polynomially bounded, polynomial time recognizable and characterizes L by $L = \{x : \exists y \, s.t. \, (x, y) \in R_L\}$*

We say that y is a witness for the membership $x \in L$ if $(x, y) \in R_L$. We will also let $R_L(x)$ denote the set of witnesses for the membership $x \in L$, i.e., $R_L(x) = \{y : (x, y) \in L\}$. In the following, we assume a fixed witness relation R_L for each language $L \in \mathcal{NP}$.

A.2 Indistinguishability

Definition 4 (Computational Indistinguishability)**.** *Let Y be a countable set. Two ensembles $\{A_{n,y}\}_{n\in N, y\in Y}$ and $\{B_{n,y}\}_{n\in N, y\in Y}$ are said to be* computationally indistinguishable *(denoted by $\{A_{n,y}\}_{n\in N, y\in Y} \approx \{B_{n,y}\}_{n\in N, y\in Y}$), if for every $\mathcal{PPT}$ "distinguishing" machine D, there exists a negligible function $\nu(\cdot)$ so that for every $n \in N, y \in Y$:*

$$|\Pr[a \leftarrow A_{n,y} \ : \ D(1^n, y, a) = 1] - \Pr[b \leftarrow B_{n,y} \ : \ D(1^n, y, b) = 1]| < \nu(n)$$

A.3 Interactive Proofs

We use the standard definitions of interactive proofs (and interactive Turing machines) [GMR89] and arguments (a.k.a computationally-sound proofs) [BCC88]. Given a pair of interactive Turing machines, P and V, we denote by $\langle P(w), V\rangle(x)$ the random variable representing the (local) output of V, on common input x, when interacting with machine P with private input w, when the random input to each machine is uniformly and independently chosen.

Definition 5 (Interactive Proof System)**.** *A pair of interactive machines $\langle P, V\rangle$ is called an* interactive proof system *for a language L if there is a negligible function $\nu(\cdot)$ such that the following two conditions hold :*

- Completeness: *For every $x \in L$, and every $w \in R_L(x)$,* $\Pr[\langle P(w), V\rangle(x) = 1] = 1$
- Soundness: *For every $x \notin L$, and every interactive machine B,* $\Pr[\langle B, V\rangle(x) = 1] \leq \nu(|x|)$

In case that the soundness condition is required to hold only with respect to a computationally bounded prover, the pair $\langle P, V\rangle$ is called an interactive argument *system.*

A.4 Witness Indistinguishable Proofs

The notion of *witness indistinguishability* ($\mathcal{WI}$) was introduced by Feige and Shamir in [FS90]. Roughly speaking, an interactive proof is said to be $\mathcal{WI}$ if the verifier's output is "computationally independent" of the witness used by the prover for proving the statement. In this context, we focus on languages $L \in \mathcal{NP}$ with a corresponding witness relation R_L. Namely, we consider interactions in which, on common input x, the prover is given a witness in $R_L(x)$. By saying that the output is computationally independent of the witness, we mean that for any two possible $\mathcal{NP}$-witnesses that could be used by the prover to prove the statement $x \in L$, the corresponding outputs are computationally indistinguishable.

Definition 6 (Witness-indistinguishability)**.** *Let $\langle P, V\rangle$ be an interactive proof system for a language $L \in \mathcal{NP}$. We say that $\langle P, V\rangle$ is* witness-indistinguishable *for R_L, if for every $\mathcal{PPT}$ ITM V^* and for every two sequences $\{w^1_{n,x}\}_{n\in N, x\in L\cap\{0,1\}^n}$ and $\{w^2_{n,x}\}_{n\in N, x\in L\cap\{0,1\}^n}$, such that $w^1_{n,x}, w^2_{n,x} \in R_L(x)$ for every x, the following probability ensembles are computationally indistinguishable.*

- $\{\langle P(w^1_{n,x}), V^*(z)\rangle(x)\}_{n\in N, x\in L\cap\{0,1\}^n, z\in\{0,1\}^*}$
- $\{\langle P(w^2_{n,x}), V^*(z)\rangle(x)\}_{n\in N, x\in L\cap\{0,1\}^n, z\in\{0,1\}^*}$

A.5 Special-sound $\mathcal{WI}$ proofs

A 4-round public-coin interactive proof for the language $L \in \mathcal{NP}$ with witness relation R_L is special-sound with respect to R_L, if for any two transcripts $(\delta, \alpha, \beta, \gamma)$ and $(\delta', \alpha', \beta', \gamma')$ such that the initial two messages, δ, δ' and α, α', are the same but the challenges β, β' are different, there is a deterministic procedure to extract the witness from the two transcripts and runs in polynomial time. In this paper, we rely on special sound proofs that are also witness indistinguishable ($\mathcal{WI}$) Special-sound $\mathcal{WI}$ proofs for languages in $\mathcal{NP}$ can be based on the existence of 2-round commitment schemes, which in turn can be based on one-way functions [GMW91, FS90, HILL99, Nao91].

A.6 Concurrent $\mathcal{ZK}$ Protocols

Let $\langle P, V\rangle$ be an interactive proof for a language L. Consider a concurrent adversarial verifier V^* that on common input a security parameter 1^n, a statement $x \in \{0,1\}^n$ and auxiliary input z, interacts with $m(n)$ independent copies of P concurrently, without any restrictions over the scheduling of the messages in the different interactions with P.

Definition 7. *Let $\langle P, V\rangle$ be an interactive proof system for a language L. We say that $\langle P, V\rangle$ is* black-box concurrent zero-knowledge *if for every polynomials q and m, there exists a probabilistic polynomial time algorithm $S_{q,m}$, such that for every concurrent adversary V^* that on common input 1^n, x and auxiliary input z opens up $m(n)$ executions and has a running-time bounded by $q(n)$, $S_{q,m}(1^n, x, z)$ runs in time polynomial in n. Furthermore, it holds that the following ensembles are computationally indistinguishable*

- $\{\mathsf{view}_{V^*}[\langle P(w), V^*(z)\rangle(1^n, x)]\}_{n\in N, x\in L\cap\{0,1\}^n, w\in R_L(x), z\in\{0,1\}^*}$
- $\{S_{q,m}(1^n, x, z)\}_{n\in N, x\in L\cap\{0,1\}^n, w\in R_L(x), z\in\{0,1\}^*}$

A.7 Extractable Commitments

We recall the definition of extractable commitments defined in [PW09]. Let $\langle C_s, R_s\rangle$ be a statistically binding commitment scheme. We say that $\langle C_s, R_s\rangle$ is an extractable commitment scheme if there exists an expected polynomial-time probabilistic oracle machine (the extractor) E that given oracle access to any $\mathcal{PPT}$ cheating sender C^* outputs a pair (τ, σ^*) such that:

- (simulation) τ is identically distributed to the view of C^* at the end of interacting with an honest receiver R_s in commit phase.
- (extraction) the probability that τ is *accepting and valid* $\sigma^* = \perp$ is negligible.
- (binding) if $\sigma^* \neq \perp$, then it is statistically impossible to open τ to any value other than σ^*.

We will also consider extractable commitment schemes that are computationally binding; the definition is as above, except if $\sigma^* \neq \perp$, we only require that it is computationally infeasible to open τ to any value other than σ^*.

A.8 Trapdoor Commitments

We say that $\langle C_t, R_t \rangle$ is a trapdoor commitment scheme if there exists an expected polynomial-time probabilistic oracle machine $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PPT R^* and all $v \in \{0,1\}^n$, the output (τ, w) of the following experiments are computationally indistinguishable:

- an honest sender C_t interacts with R^* to commit to v, and then opens the commitment: τ is the view of R^* in the commit phase, and w is the message C_t sends in the open phase.
- the simulator $\mathcal{S}$ generates a simulated view τ for the commit phase, and then opens the commitment to v in the open phase: formally, $\mathcal{S}_1^{R^*}(1^n, 1^k) \to (\tau, \text{STATE}), \mathcal{S}_2(\text{STATE}, v) \to w$.

B Proofs for Our Black-Box Robust CCA-Secure Commitments

B.1 Properties of the Trapdoor Commitments TrapCom

Before proving the security properties of our robust CCA-secure commitment scheme, we first prove a few properties of the trapdoor commitment scheme TrapCom of [PW09], which will be very instrumental the proof of robust CCA-security.

Special-soundness of TrapCom: Since Stage 2 of the protocol TrapCom is simply a PExtCom commitment, given any two *admissible transcripts* of Stage 2, a committed value can be extracted. Consider the following deterministic polynomial time procedure reconst that on input two *admissible transcripts* $\mathcal{T}_1$, $\mathcal{T}_2$ of Stage 2 extracts a committed value as follows: It first reconstructs all the matrices $v_1, \cdots, v_n$ from $\mathcal{T}_1, \mathcal{T}_2$ by relying on the extractability of PExtCom; then it checks whether all the left columns of the matrices sum up to the same bit b, and sets σ_0 to b if this is the case and $\perp$ otherwise; it computes σ_1 similarly with respect to the right columns; next,

- If σ_1 and σ_1 equal to two different bits, reconst outputs err.
- Else if σ_0 and σ_1 both equal to $\perp$, it outputs $\perp$ as well.
- Finally, if σ_0 and σ_1 equal to the same bit b, or only one of them equals to b and the other equals to $\perp$, reconst outputs b.

We show below in Lemma 3 that as long as the reconst procedure does not output err, then the extracted value must be the committed value—we call this the special-soundness property of TrapCom—and in Lemma 4 that when the reconst procedure outputs err, then the receiver's challenge in the TrapCom commitment can be computed efficiently. It follows essentially from Lemma 3 and 4 that TrapCom is computationally hiding. Formally, let $\mathcal{T}$ be a commitment of TrapCom, we say that a pair of admissible transcripts $\mathcal{T}_1, \mathcal{T}_2$ is consistent with TrapCom if the first message in $\mathcal{T}_1$ and $\mathcal{T}_2$ equals to the first message of Stage 2 in $\mathcal{T}$. Then,

Lemma 3 (Special-soundness of TrapCom)**.** *Let $\mathcal{T}$ be a commitment of* TrapCom*, and $\mathcal{T}_1, \mathcal{T}_2$ a pair of admissible transcripts of* TrapCom *that is consistent with $\mathcal{T}$. Then, if* $\mathsf{reconst}(\mathcal{T}_1, \mathcal{T}_2) = \sigma \neq \mathsf{err}$*, it is statistically impossible to open $\mathcal{T}$ to any value other than σ.*

Proof. Assume for contradiction that $\mathsf{reconst}(\mathcal{T}_1, \mathcal{T}_2)$ outputs $\sigma \neq \mathsf{err}$ but there exists a decommitment that opens $\mathcal{T}$ to a bit $\sigma' \in \{0,1\}$ different from σ.

It follows from the validity condition of TrapCom that if a commitment can be opened to σ' there must exist a $\gamma \in \{0,1\}$, such that all the commitments of ExtCom to $v_i^{0\gamma}, v_i^{1\gamma}$ for $i = 1, 2, \ldots, k$ are valid and $\sigma' = v_1^{0\gamma} \oplus v_1^{1\gamma} = \cdots = v_k^{0\gamma} \oplus v_k^{1\gamma}$, where σ is the committed value. That is, either all the left columns or the right columns are valid commitments to values that sum up to σ'. Since two admissible transcripts of TrapCom contains a pair of admissible transcripts of ExtCom for each commitment to a bit $v_j^{b_1 b_2}$. It follows from the extractability of ExtCom that from $\mathcal{T}_1$ and $\mathcal{T}_2$, values $v_i^{0\gamma}, v_i^{1\gamma}$ can be extracted correctly, as by the validity condition commitments to $v_i^{b\gamma}$'s are all valid. Therefore the procedure reconst will set bit σ_γ to σ'. Then, conditioned on reconst does not output err, it must output σ', which gives a contradiction. □

Lemma 4. *Let $\mathcal{T}$ be an* accepting *commitment of* TrapCom*, and $\mathcal{T}_1, \mathcal{T}_2$ a pair of admissible transcripts of* TrapCom *that is consistent with $\mathcal{T}$. Then, if* $\mathsf{reconst}(\mathcal{T}_1, \mathcal{T}_2) = \mathsf{err}$*, the receiver's challenge committed to in Stage 1 of $\mathcal{T}$ can be computed efficiently and deterministically from $\mathcal{T}_1, \mathcal{T}_2$.*

Proof. The reconst procedure, on input $\mathcal{T}_1, \mathcal{T}_2$, reconstructs a tuple of n matrices $\tilde{v}_1, \ldots, \tilde{v}_n$ by relying on the special soundness of ExtCom, and outputs err if and only if all the values $\tilde{v}_j^{00}, \tilde{v}_j^{10}$ in the left columns sum up to a bit b whereas all the values $\tilde{v}_j^{01}, \tilde{v}_j^{11}$ in the right columns sum up to $1 - b$. Furthermore, since $\mathcal{T}$ is accepting, in Stage 3 of $\mathcal{T}$, the receiver must successfully open the stage 1 com commitment to a challenge e and the committer must successfully open the two commitments in the e_j^{th} row of the j^{th} matrices to the same value η_j for every $j \in [k]$. Thus, by the special soundness of ExtCom, values extracted from the e_j^{th} row $v_j^{e_j 0}, v_j^{e_j 1}$ must equal to η_j, which means the two bits $v_j^{(1-e_j)0}, v_j^{(1-e_j)1}$ extracted from the $(1-e_j)^{\text{th}}$ row must differ. Thus from $\mathcal{T}_1$ and $\mathcal{T}_2$, the receiver's challenge e in $\mathcal{T}$ can be computed efficiently. □

Strong Computational Binding: We show that TrapCom enjoys a strong computational binding property as described in Lemma 5.

Lemma 5 (Strong computational binding)**.** *For every $\mathcal{PPT}$ machine C^*, the probability that C^* in interaction with the honest receiver of* TrapCom*, generates a commitment that has more than one valid committed values is negligible.* [4]

Proof. Assume for contradiction that there is a committer C^* that can generate a commitment that has two valid committed values with polynomial probability. Then we can construct a machine A that violates the hiding property of com.

The machine A on input a com commitment c to a random n-bit string e, internally incorporates C^* and emulates the messages from the receiver for C^* honestly, except that: it forwards c to A at the Stage 1 message; after C^* completes Stage 2, it repeatedly rewinds C^* from the challenge message in Stage 2 by sending C^* freshly sampled challenges, until another accepting transcript of Stage 2 is obtained; then it checks whether the pair of transcripts of Stage 2 is admissible and if so whether reconst outputs err on input these two transcripts; if this is the case, it computes the challenge e' from the two admissible transcripts by Lemma 4 and outputs e'; otherwise, it aborts. Finally, A cuts its execution off after $2^{n/3}$ steps.

[4] Note that the computational binding property only guarantees that it is impossible for an efficient committer to generate a commitment of TrapCom and opens it to two different values.

It follows from standard technique that the expected running time of C^* is bounded by a polynomial. Furthermore, each challenge in Stage 2 of a $\mathsf{TrapCom}$ commitment is a n-tuple of n-bit strings. Then Since A runs for at most $2^{n/3}$ steps, the probability that any n-bit string is picked for a second time in A is $2^{n/3}/2^n$; since A picks at most $2^{n/3}$ strings, by union bound, the probability that any n-bit string is picked twice is $\frac{1}{2^{n/3}} = 2^{n/3}\frac{2^{n/3}}{2^n}$. Therefore, except with negligible probability, the pair of accepting transcripts collected by A is also admissible.

Next, we show that conditioned on that the pair of transcripts collected by A is admissible, A outputs the value committed to in c with polynomial probability. Since A emulates the execution of C^* perfectly, with polynomial probability C^* in A generates a commitment that can be opened to both 0 and 1. When this happens, by the validity condition of $\mathsf{TrapCom}$, the commitment generated by C^* must have the property that all the commitments of ExtCom in Stage 2 are valid, and the committed values in the left columns sum up to a bit b whereas the committed values in the right columns sum up to another bit $1-b$. In this case, the procedure $\mathsf{reconst}$ fails to extract a value from the pair of admissible transcripts collected by A and outputs err. The by Lemma 4 the challenge committed to in Stage 1 can be computed. Thus A outputs the committed value of c with polynomial probability. □

Extension to multiple bits. As shown in [PW09], by running the trapdoor bit commitment scheme $\mathsf{TrapCom}$ in parallel, we obtain a trapdoor commitment scheme $\mathsf{PTrapCom}$ for multiple bits, with the additional property that we can open the commitment to any subset of the bits without compromising the security of the remaining bits. The hiding, binding and trapdoor property of the commitment remains. Furthermore, Stage 2 of the protocol $\mathsf{PTrapCom}$ consists of many parallel executions of $\mathsf{PExtCom}$. We say that two transcripts of Stage 2 of $\mathsf{PTrapCom}$ are $\mathsf{admissible}$ if they contain a pair of admissible transcripts of $\mathsf{PTrapCom}$ for each parallel execution in it. Then given a pair of admissible transcripts of $\mathsf{PTrapCom}$, the committed string can be extracted by running the following procedure $\overline{\mathsf{reconst}}$: For each parallel execution, it extracts a value σ_i using the $\mathsf{reconst}$ procedure; then it outputs err if any σ_i equals to err, otherwise, it outputs all the extracted bits σ_i's concatenated. Again, we call this property the special-soundness of $\mathsf{PTrapCom}$.

B.2 $\langle C, R\rangle$ is a Statistically-Binding Commitment Scheme

In this section, we provide formally that that $\langle C, R\rangle$ is a statistically binding commitment scheme.

Proposition 2. *$\langle C, R\rangle$ is a statistically binding commitment scheme.*

Proof. The statistically bindng property of $\langle C, R\rangle$ follows directly from that of com. We then, focus on showing the hiding property.

Recall that the commitment scheme $\mathsf{TrapCom}$ is trapdoor. In particular, as shown in [PW09], if the receiver's challenge is fixed, then there is a straight-line simulator that can generate a simulated transcript of the commit phase that can be equivocated to both 0 and 1 later. We recall the simulation strategy. Let R^* be a malicious receiver using a fixed challenge e, then to simulate Stage 2 and 3 of $\mathsf{TrapCom}$ for R^*, the simulator samples a random bit γ and prepares $v_1 \cdots v_n$ where each v_i is a 2×2 0,1-matrix such that, the e_i^{th} row of v_i has the form (η_i, η_i) and the $(1-e_i)^{\text{th}}$ row has the form $(\gamma \oplus \eta_i, (1-\gamma) \oplus \eta_i)$ with a randomly and independently sampled bit η_i. It then commits to $v_1, \cdots, v_n$ using $\mathsf{PExtCom}$ in Stage 2; later, in Stage 3, upon receiving challenge e, for every i, it opens commitments to the e_i^{th} row in the i^{th} matrix to (η_i, η_i), yielding an accepting commitment. To equivocate the simulated commitment to 0, the simulator sends γ and opens all the commitments in the γ^{th} column of the matrices; to equivocate the commitment to 1, it sends $1-\gamma$ and opens the $(1-\gamma)^{\text{th}}$ column of the matrices. It follows from the hiding property of ExtCom (recall that a commitment of $\mathsf{PExtCom}$ to $v_1 \cdots v_n$ is simply many commitments of ExtCom to bits

$v_j^{b_1,b_2}$ in parallel) that, for every $b \in \{0,1\}$, the simulated commitment of TrapCom together with the equivocated opening to b is indistinguishable from an honest commitment and opening to b. Furthermore, since the simulation is straight-line and thus is composable under concurrent composition, we have that TrapCom is secure under selective opening attack with concurrent composition (See [Xia11] for a formal definition) against malicious receivers that always use a fixed challenge.

Next, by relying on the security against selective opening attack of TrapCom, we show that for every malicious receiver R^* of $\langle C, R\rangle$, there is a simulator S that can generate a simulated commitment that is indistinguishable from an honest commitment to R^* to any value v; then the hiding property follows. More precisely, given a malicious receiver R^* of $\langle C, R\rangle$ (with loss of generality, deterministic), let c_1 be the Stage 1 commitment from R^* and e the challenge committed to in c_1. The simulator S, on input e, simulates a commitment of $\langle C, R\rangle$ to v as follows: In Stage 2 and 3, it simulates the $10nL(n)$ commitments of PTrapCom*w.r.t. challenge* e by using the simulator of PTrapCom described above does; finally in Stage 4, upon receiving Γ, for every column $j \in \Gamma$, it samples a random string $\tilde{s}_j$ and equivocate all the simulated commitments of PTrapCom in the j^{th} column to $\tilde{s}_j$. Since R^* uses a fixed challenge e in all the PTrapCom commitments, it follows from the security against selective opening attack of TrapCom that in H the simulated commitments of PTrapCom in Stage 2 and 3, together with the equivocated openings to n random values in Stage 4 is indistinguishable from, the honest commitments and openings to n shares of v in the real execution (since by the property of the $(n+1)$-out-of-$10n$ secret-sharing, n shares of v is identically distributed to n random values). Thus, the simulated commitment by S is indistinguishable to an honest commitment to v. Since S does not depend on the committed value v, we conclude that honest commitments to any two values v_1 and v_2 are indistinguishable. □

B.3 Proof of Robust CCA-Security of $\langle C, R\rangle$

In this section, we prove the following theorem.

Theorem 2. *$\langle C, R\rangle$ is $\kappa(n)$-robust CCA-secure w.r.t. committed value oracles.*

The formal proof of Theorem 2 consists of two parts: in Section B.3.2, we show that $\langle C, R\rangle$ is CCA-secure. and in section B.3.3, we show that it is also robust. Towards this, below we first adapt the definition of safe-points in [CLP10] to work with our protocol $\langle C, R\rangle$.

B.3.1 Safe-Points

Our notion of safe-points is almost the same as that in [CLP10] (which in turn is based on the notion of safe-points of [LPV08] and safe rewinding block of [DDN00]), with the only exception that our definition considers the 3-round rows in our black-box construction of CCA commitment $\langle C, R\rangle$, instead of the 3-round $\mathcal{WISSP}$ proofs in the non-black-box construction in [CLP10]. Recall that, like a $\mathcal{WISSP}$ proof, each row in Stage 3 of the protocol $\langle C, R\rangle$ has the 3-round challenge-reply structure—we call the three messages respectively, the commit, challenge and reply messages—and has the property that rewinding a complete row reveals nothing about the committed value.

Let Δ be a transcript of one left and many right interactions of $\langle C, R\rangle$. Intuitively, a safe-point ρ of a right interaction k is a prefix of a transcript Δ that lies in between the first two messages α_r and β_r of a row $(\alpha_r, \beta_r, \gamma_r)$ in interaction k, such that, when rewinding from ρ to γ_r, if A uses the *same* "scheduling of messages" as in Δ, then the left interaction can be emulated without affecting the hiding property. This holds, if in Δ from ρ to where γ_r is sent, A expects either no message or only complete rows in the left interaction, as shown in Figure 4 (i) and (ii) respectively, (Additionally, in both cases, A may request the reply message of some row in the left, as shown in Figure 4 (iii). This is because, given the first two messages of a row, the reply message is deterministic, and hence can be emulated in the rewinding by replaying the reply in Δ.)

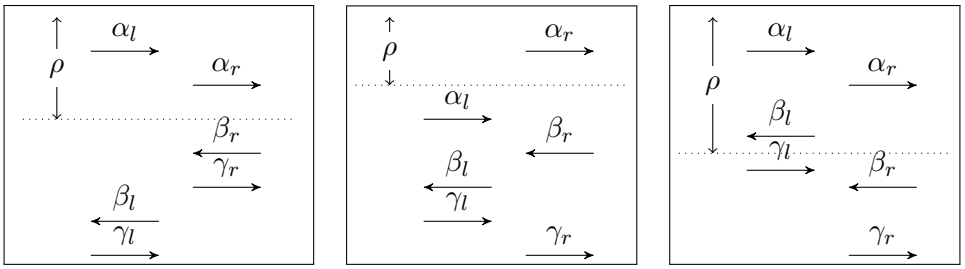

Figure 4: Three characteristic safe-points.

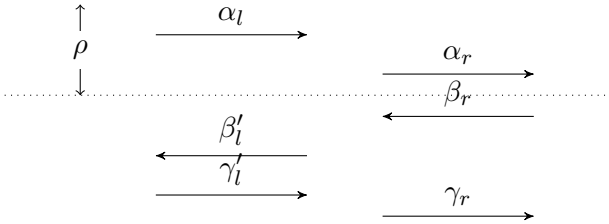

Figure 5: Prefix ρ that is not a safe point.

Definition 8. *Let Δ be any transcript of one left interaction, and many right interactions, of $\langle C, R\rangle$. A prefix ρ of a transcript Δ is called a* safe-point *for right interaction k, if there exists an accepting row $(\alpha_r, \beta_r, \gamma_r)$ in the right interaction k, such that:*

1. *α_r occurs in ρ, but not β_r (and γ_r).*
2. *for any row $(\alpha_l, \beta_l, \gamma_l)$ in the left interaction, if α_l occurs in ρ, then β_l occurs after γ_r.*
3. *messages in Stage 1, 3, and 4 of the left interaction occur either before ρ or after γ_r.*

If ρ is a safe-point, let $(\alpha_\rho, \beta_\rho, \gamma_\rho)$ denote the canonical "safe" right row associated with ρ. Note that the only case a right-interaction row is not associated with any safe-point is if it is "aligned" with a left-execution row, as shown in Figure 5. In contrast, in all other cases, a right-interaction row has a safe-point, as shown in Figure 4.

It follows from exactly the same proof in [CLP10] that in any transcript of one left and many right interactions of $\langle C, R\rangle$, every accepting right interaction that has a different identity from the left interaction, has at least $\eta(n)$ safe-points. This technical lemma will be very instrumental in the proof of CCA-security in the next section.

Lemma 6 (Safe-point Lemma)**.** *Let Δ be any transcript of one left interaction, and many right interactions, of $\langle C, R\rangle$. Then, in Δ, for every successful right interaction that has a different identity from the left interaction, there exist at least a number of $\Omega(\eta(n))$* non-overlapping *rows that are associated with a* safe-point*.*

B.3.2 Proof of CCA Security

We show that for every $\mathcal{PPT}$ adversary A, the following ensembles are computationally indistinguishable.

- $\{\mathsf{IND}_0(\langle C, R\rangle, A, n, z)\}_{n\in N, z\in\{0,1\}^*}$

- $\{\mathsf{IND}_1(\langle C, R\rangle, A, n, z)\}_{n\in N, z\in\{0,1\}^*}$

Towards this, we consider new commitment scheme $\langle \hat{C}, \hat{R}\rangle$ (similar to the "adaptor" schemes of [DDN00, LPV08, CLP10]), which is a variant of $\langle C, R\rangle$ where the receiver can ask for an arbitrary number of designs in Stage 2. Furthermore, $\langle \hat{C}, \hat{R}\rangle$ does not have a fixed scheduling in Stage 2; the receiver instead gets to choose which design to execute in each iteration (by sending bit b to select design_b). Note that, clearly, any execution of $\langle C, R\rangle$ can be emulated by an execution of $\langle \hat{C}, \hat{R}\rangle$ by simply requesting the appropriate designs. It follows using essentially the same proof for the hiding property of $\langle C, R\rangle$ in Proposition 2 that $\langle \hat{C}, \hat{R}\rangle$ is computationally hiding; we omit the proof here.

Now assume for contradiction that there exists an adversary A, a distinguisher D, and a polynomial p, such that for infinitely many $n \in N$, there exists $z \in \{0,1\}^*$, such that,

$$\left| \Pr\left[D(\mathsf{IND}_0(\langle C, R\rangle, A, n, z)) = 1\right] - \Pr\left[D(\mathsf{IND}_1(\langle C, R\rangle, A, n, z)) = 1\right] \right| \geq \frac{1}{p(n)}$$

We reach a contradiction by exhibiting a (stand-alone) adversary B^* that distinguishes commitments using $\langle \hat{C}, \hat{R}\rangle$. Let $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)$ denote the output of $B^*(1^n, z)$ after receiving a commitment of $\langle \hat{C}, \hat{R}\rangle$ to value v_b, where as in experiment IND_b the challenges v_0 and v_1 are chosen adaptively by B^*. We show that the following two claims hold w.r.t B^*.

Lemma 7. *There exists a polynomial function t and a negligible function μ, such that for every $b \in \{0,1\}$, $n \in N$ and $z \in \{0,1\}^*$, and every function p, the probability that B^* in experiment $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)$ takes more than $p(n)$ steps is less than or equal to $\frac{t(n)}{p(n)} + \mu(n)$.*

Lemma 8. *Let $b \in \{0,1\}$. The following ensembles are computationally indistinguishable.*

- $\left\{\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)\right\}_{n\in N, z\in\{0,1\}^*}$
- $\left\{\mathsf{IND}_b(\langle C, R\rangle, A, n, z)\right\}_{n\in N, z\in\{0,1\}^*}$

By Lemma 8, it thus follows that for infinitely many $n \in N$, there exists $z \in \{0,1\}^*$, such that,

$$\left|\Pr\left[D(\mathsf{STA}_0(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)) = 1\right] - \Pr\left[D(\mathsf{STA}_1(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)) = 1\right]\right| \geq \frac{3}{4p(n)}$$

Furthermore, by Lemma 7, the probability that B^* runs for more than $T(n) = 4t(n)p(n)$ steps is smaller than $1/4p(n)$. Therefore, the execution of B^* can be truncated after $T(n)$ steps, while only affecting the distinguishing probability by at most $\frac{1}{4p(n)}$, which means there exists a $\mathcal{PPT}$ machine that distinguishes commitments with probability $\frac{1}{2p(n)}$; this contradicts the hiding property of $\langle \hat{C}, \hat{R}\rangle$.

Construction of B^*. On a high-level, B^* in interaction with an honest committer $\hat{C}$ on the left emulates the committed-value oracle $\mathcal{O}$ for A by extracting the committed values of the right interactions from the rows in Stage 2 of $\langle C, R\rangle$. Recall that Stage 2 of $\langle C, R\rangle$ contains multiple rows; each in turn contains commitments to secret shares of the committed value (more precisely, shares of a decommitment of Stage 2 of $\langle C, R\rangle$), using Stage 2 of $\mathsf{PTrapCom}$. It follows from the special soundness of $\mathsf{PTrapCom}$ that, given a pair of transcripts of a row that are admissible for each commitment of $\mathsf{TrapCom}$ contained in that row, the secret shares committed in that row can be reconstructed using the $\overline{\mathsf{reconst}}$ procedure (provided that it does not output err)—we say that such a pair of transcripts of a row is admissible. Then the committed value can be recovered.

The construction of B^* contains two parts, a rewinding procedure and a reconstruction procedure. The rewinding procedure recursively rewinds the rows of the right integrations and guarantees that at the end of every accepting right interaction with different identity from the left interaction, a pair of admissible transcripts (of one row) of that right interaction is collected. The reconstruction procedure, on the other hand, on input a pair of admissible transcripts of one right interaction, reconstructs the committed value of that interaction. The rewinding procedure that we use here is essentially the same as that used in the CLP extraction procedure, except from the superficial difference that in [CLP10], pairs of accepting transcripts of $\mathcal{WISSP}$ proofs are collected. However, in [CLP10] given two different accepting transcripts of a $\mathcal{WISSP}$ proof in one right interaction, a committed value can be extracted directly using the special-soundness property *without over-extraction*. In this work, in order to avoid the over-extraction problem, the procedure for reconstructing the committed value from two admissible transcripts is much more involved; we employ the technique used in [CDSMW08, CDSMW09, Wee10] and formalize it in the reconstruction procedure REC.

Next, we formally describe the rewinding procedure, which invokes the reconstruction procedure REC whenever it obtains a pair of admissible transcripts; readers who are familiar with the CLP extraction procedure can skip this part and jump to the description reconstruction procedure REC.

The Rewinding Procedure At a high-level, the rewinding procedure rewinds A only from safe-points. This ensures that we do not have to rewind the external left execution; rather, it suffices to request an additional design on the left to handle these rewindings. But, as the simulator needs to provide the committed values in a "on-line" fashion (i.e., as soon as a right-interaction completes, the simulator needs to provide A with decommitment information for this interaction), these rewindings might become recursive (if the right interactions are nested). And, if we were to perform these rewindings naively, the running-time quickly grows exponentially (just as in the context of *concurrent zero knowledge* [DNS04]). To make sure that the recursion depth is constant, we instead only rewind from safe-points ρ such that the number of new right-rows that start between ρ and the last message γ_ρ of the right-row associated with ρ, is "small"; here, "small" is defined appropriately based on the recursion level. More precisely, we say that a safe-point ρ is $d+1$-*good* for a transcript Δ if less than $k_d = M/\eta'^d$ right-rows start between ρ and γ_ρ, where M is an upper-bound on the total number of messages that A sends or receives, and $\eta' = n^{\varepsilon'}$ for some constant ε' such that $0 < \varepsilon' < \varepsilon$. On recursion level d, B^* then only rewinds A from $d+1$-good safe-points.

Formally, we describe the rewinding procedure using a recursive helper procedure EXT. EXT, on input an integer d (the recursion level), a partial joint view $\mathcal{V}$ of A and the (emulated) right receivers, the index s of a right-row, a "repository" $\mathcal{R}$ of pairs of admissible transcripts of the right interactions that have been previously collected, proceeds as follows.

PROCEDURE EXT$(d, \mathcal{V}, s, \mathcal{R})$: Let ρ be the (partial) transcript contained in $\mathcal{V}$. If $d = 0$, EXT will emulates a complete execution of IND with the adversary A. If $d > 0$, it will instead extends the partial view $\mathcal{V}$ to the completion of the right-row s; if at any point in the emulation, ρ is not a $d+1$-good safe-point for s, EXT aborts and returns $\perp$. Finally, EXT returns the the view $\mathcal{V}_A$ of A in the emulation (generated so far). We now turn to describe how EXT emulates the left and the right interactions.

The left interaction is emulated by simply requesting the appropriate messages from the external committer. At the top level (i.e., $d = 0$), A participates in a *complete* $\langle C, R\rangle$ commitment on the left, which can be easily emulated by simply requesting the appropriate designs from $\hat{C}$. At lower levels (i.e., $d > 0$), EXT cancels every execution in which ρ is not a safe-point. Hence it only needs to emulate the left interaction when ρ is a safe-point. In this case, as previously discussed, A either does not request any new messages on the left, or only asks for complete new rows; the former case can be trivially emulated (by simply doing nothing or replaying old messages if A asks for the

third message of a left row again), in the latter case, EXT emulates the left interaction by asking for more designs from $\hat{C}$.

On the other hand, in the right interactions EXT follows the honest receiver strategy of $\langle C, R\rangle$. Furthermore, whenever A completes a row $(\alpha_r, \beta_r, \gamma_r)$ in a right interaction j, EXT attempts to extract a decommitment for this interaction, if the row $(\alpha_r, \beta_r, \gamma_r)$ is associated with a $d+1$-good safe-point ρ'. To extract, EXT invokes itself recursively on input $(d+1, \mathcal{V}', s', \mathcal{R})$, where $\mathcal{V}'$ is the (partial) joint view of A and the right receivers corresponding to the transcript ρ', and s' is the index of the right-row $(\alpha_r, \beta_r, \gamma_r)$. It continues invoking itself recursively until one of the recursive invocations returns a view containing another accepting transcript $(\alpha_r, \beta'_r, \gamma'_r)$ of the s'^{th} row. When this happens, if $(\alpha_r, \beta_r, \gamma_r)$ and $(\alpha_r, \beta'_r, \gamma'_r)$ are a pair of admissible transcripts, EXT records them in the repository $\mathcal{R}$. Later, whenever A expects the committed value for a right interaction j, it simply checks the repository $\mathcal{R}$ for a matching pair of admissible transcripts $\mathcal{T}_1, \mathcal{T}_2$, and invokes REC with $\mathcal{T}_1, \mathcal{T}_2$ and the transcript $\mathcal{T}$ of the right integration j to obtain the committed value u; it aborts and outputs fail if no pair of admissible transcripts is available or the REC procedure returns err—we say that EXT "gets stuck" on interaction j in this case. (If A expects the committed value of a right interaction that fails or has the same identity as the left, it simply sends $\perp$ to A.)

The Reconstruction Procedure Let $\mathcal{T}_1 = (\alpha_r, \beta_r, \gamma_r)$ and $\mathcal{T}_2 = (\alpha_r, \beta'_r, \gamma'_r)$ be a pair of admissible transcripts of one row of a right commitment $\mathcal{T}$. Recall that Stage 2 in $\mathcal{T}$ contains a com commitment to the committed value v, and each row in $\mathcal{T}$ commits to the $10n$ secret shares of a decommitment (v, d) of the com commitment, using Stage 2 of $\mathsf{PTrapCom}$. Since $\mathcal{T}_1$ and $\mathcal{T}_2$ are admissible, they contain a pair of admissible transcripts of $\mathsf{PTrapCom}$ for each commitment to one of the $10n$ share. Therefore, by the special-soundness property of $\mathsf{PTrapCom}$, a value can be reconstructed from each pair of admissible transcripts of $\mathsf{PTrapCom}$ using the $\overline{\mathsf{reconst}}$ procedure; if the extracted value is not err, then either the corresponding commitment is invalid, or it is and the reconstructed value is the committed share. At a high-level, the reconstruction procedure REC uses this property to try to extract shares of the decommitment (v, d) and then decode the shares to obtain the committed value; it utilizes the final cut-and-choose Stage in the right commitment $\mathcal{T}$ to avoid over-extraction. Formally,

PROCEDURE $\mathsf{REC}(\mathcal{T}_1, \mathcal{T}_2, \mathcal{T})$: For every $j \in [10n]$, REC sets T_1^j, T_2^j to the pair of admissible transcripts of $\mathsf{PTrapCom}$ for the commitment to the j^{th} secret share contained in $\mathcal{T}_1, \mathcal{T}_2$, and y_j to the output of $\overline{\mathsf{reconst}}(T_1^j, T_2^j)$; it aborts and outputs err if y_j equals to err; otherwise, it sets the j^{th} share $\tilde{s}_j$ to y_j. After extracting all the shares $\tilde{\pi} = (\tilde{s}_1, \ldots, \tilde{s}_{10n})$, it recovers a valid codeword w that is 0.8-*close* to $\tilde{\pi}$; then, it checks whether w agrees with all the shares revealed in the cut-and-choose stage in the right commitment $\mathcal{T}$ (that is, for every $i \in [\Gamma]$, it checks whether w_i equals to the share $\hat{s}_i$ revealed for the i^{th} column in the cut-and-choose stage in $\mathcal{T}$); if this holds, it decodes w to a tuple (v', d'), and outputs v' if (v', d') is a valid decommitment of the Stage 2 com commitment in $\mathcal{T}$. It aborts and outputs $\perp$ whenever any of the following events happens: (1) the extracted shares $\tilde{\pi}$ is not 0.8-close to any valid codeword, or (2) the codeword w does not agree with any of the shares revealed in the cut-and-choose stage, or (3) the tuple (v', d') decoded from w is not a valid decommitment of the Stage 2 com commitment.

We now return to the description of B^*. B^* in interaction with $\hat{C}$, simply invokes EXT on inputs $(0, \mathcal{V}, \mathsf{null}, \emptyset)$, where $\mathcal{V}$ is the initial joint states of A and honest right receivers. Once EXT returns a view $\mathcal{V}_A$ of A, B^* return the output of A in this view if A never used the identity of the left interaction in any of the right interactions, and returns $\perp$ otherwise. Furthermore, to simplify our analysis, B^* cuts-off EXT whenever it runs for more than 2^n steps. If this happens, B^* halts and outputs fail.

Proof of Lemma 7 and 8: Next we proceed to show that B^* runs in expected polynomial time (Lemma 7) and the output of B^* is correctly distributed (Lemma 8).[5] Towards this, we consider another machine $\tilde{B}$ that proceeds identically to B^* except that it has access to an oracle $\tilde{\mathcal{O}}$ that on input an *accepting transcript* $\mathcal{T}$ of a commitment of $\langle C, R\rangle$, returns the unique committed value of that commitment if it is valid, and returns $\perp$ if it is invalid or has more than one valid committed value; furthermore, $\tilde{B}$ runs a variant $\widetilde{\mathsf{EXT}}$ of the recursive helper procedure EXT that B^* runs. $\widetilde{\mathsf{EXT}}$ proceeds identically to EXT except that whenever A during the rewindings in $\widetilde{\mathsf{EXT}}$ expects the committed value of a right commitment $\mathcal{T}$ (that is accepting and has an identity different from that of the left commitment), it queries the oracle $\tilde{\mathcal{O}}$ on $\mathcal{T}$ and feeds A the value v returned by $\tilde{\mathcal{O}}$. $\widetilde{\mathsf{EXT}}$ still reconstructs a value v' for that right interaction as EXT does, but, it does abort and outputs fail as EXT does when reconstruction fails; instead it continues the execution and simply outputs $\widetilde{\mathsf{fail}}$ on a *special output tape*; additionally, it outputs $\widetilde{\mathsf{fail}}$ on a *special output tape* if the reconstructed value v' is different from the value v returned by $\mathcal{O}^*$. Finally, as B^*, $\tilde{B}$ cuts the execution of $\widetilde{\mathsf{EXT}}$ after 2^n steps and outputs fail. Below we show that the expected running time of $\tilde{B}$ is bounded by a polynomial and the output of $\tilde{B}$ in experiment STA_b is statistically close to the view of A in the experiment IND_b.

Claim 1. *There exists a polynomial function t, such that for every $b \in \{0,1\}$, $n \in N$ and $z \in \{0,1\}^*$, $\tilde{B}$ in experiment $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, \tilde{B}^{\tilde{\mathcal{O}}}, n, z)$ takes $t(n)$ steps in expectation.*

Claim 2. *Let $b \in \{0,1\}$. The following ensembles are statistically close.*

- $\left\{\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, \tilde{B}^{\tilde{\mathcal{O}}}, n, z)\right\}_{n\in N, z\in\{0,1\}^*}$
- $\left\{\mathsf{IND}_b(\langle C, R\rangle, A, n, z)\right\}_{n\in N, z\in\{0,1\}^*}$

Furthermore, we show that except with negligible probability the, during the execution of $\tilde{B}$, the probability that $\tilde{B}$ outputs $\widetilde{\mathsf{fail}}$ on the special output tape is negligible.

Claim 3. *For every $b \in \{0,1\}$, $n \in N$ and $z \in \{0,1\}^*$, the probability that $\tilde{B}$ during the execution of $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, \tilde{B}^{\tilde{\mathcal{O}}}, n, z)$ outputs $\widetilde{\mathsf{fail}}$ on its special output tape is negligible.*

By construction, $\tilde{B}$ outputs $\widetilde{\mathsf{fail}}$ only when during the rewindings in $\widetilde{\mathsf{EXT}}$, it fails to reconstruct a committed value for a right interaction (that is accepting and has an identity different from that of the left commitment), or a value is reconstructed but is different from that returned by $\tilde{\mathcal{O}}$. By Claim 3, the above event happens with negligible probability. In other words, during the execution of $\tilde{B}$, except with negligible probability, the value $\tilde{B}$ reconstructs is always identical to that extracted by $\mathcal{O}^*$. Thus, except with negligible probability, it is equivalent to replace the values returned by the oracle $\tilde{\mathcal{O}}$ with the values $\tilde{B}$ reconstructs. Then since the only difference between $\tilde{B}$ and B^* lies in which values they use to feed the adversary A when it expects a committed value, we have that except with negligible probability, the running time and output distributions of $\tilde{B}$ are identical to that of B^*. Therefore, combining Claim 2, we directly have that for every $b \in \{0,1\}$, the output of B^* in experiment $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, \tilde{B}^{\tilde{\mathcal{O}}}, n, z)$ is statistically close to $\mathsf{IND}_b(\langle C, R\rangle, A, n, z)$, which concludes Lemma 8. Furthermore, By Claim 1, the expected running time of $\tilde{B}$ is bounded by a polynomial t; therefore, for every function T, the probability that $\tilde{B}$ runs for more than $T(n)$ steps is no more than $p(n) = t(n)/T(n)$; then, the probability that B^* runs for more than $T(n)$

[5] Though the rewinding strategy of B^* is very similar to that in [CLP10], due to the difference in the reconstruction procedure, the analysis of the running time and output distribution of B^* is quite different from that in [CLP10].

steps must be bounded by $p(n)$ plus a negligible amount, which concludes Lemma 7. Now it remains to prove Claim 1, 2 and 3.

Proof of Claim 1—Running-time Analysis of $\tilde{B}$.

To bound the expected running time of B^*, it suffices to bound the expected running time of the procedure $\widetilde{\mathsf{EXT}}$. Below in Subclaim 1 we first show that the recursive depth of $\widetilde{\mathsf{EXT}}$ is always a constant, and then bound the running time of $\widetilde{\mathsf{EXT}}$ in Subclaim 2.

Subclaim 1. *There exists a constant D such that for every $n \in N$, and every $\mathcal{V}$, s, and $\mathcal{R}$, $\widetilde{\mathsf{EXT}}(D, \mathcal{V}, s, \mathcal{R})$ does not perform any recursive calls.*

Proof. Recall that at recursion level d, the procedure $\widetilde{\mathsf{EXT}}$ terminates and returns $\perp$ whenever more than $k_d = M(n)/\eta'(n)^d$ new right-rows has started in its execution, where $M(n)$ is an upper bound on the total number of messages that the adversary A may send and receive, and $\eta'(n)$ equals to $n^{\varepsilon'}$ for some constant $0 < \varepsilon' < \varepsilon < 1$. Let n^c be an upper bound on $M(n)$; set D to $\lceil \log_{\eta'(n)} n^c \rceil$, which is a constant. When $d = D$, $k_D < 1$, which means the execution terminates whenever A starts a new right-row. On the other hand, $\widetilde{\mathsf{EXT}}$ only makes a recursive call at the completion of a new right-row. Therefore at recursion level D, $\widetilde{\mathsf{EXT}}$ never makes any recursive calls. □

Next, we show that the expected number of queries that $\widetilde{\mathsf{EXT}}$ makes to A at every recursion level $d \leq D$ is always bounded by a polynomial.

Subclaim 2. *For every $0 \leq d \leq D$, it holds that for every $n \in N$, $\mathcal{V}$, s, and $\mathcal{R}$, the expected number of queries that $\widetilde{\mathsf{EXT}}(d, \mathcal{V}, s, \mathcal{R})$ makes to A is bounded by $\theta(d) = M(n)^{3(D-d+1)}$.*

Proof. Consider some fixed $\mathcal{V}, s$ and $\mathcal{R}$. We prove the subclaim by induction on d. When $d = D$, the claim follows, since $\widetilde{\mathsf{EXT}}$ does not perform any recursive calls and the number of queries made by $\widetilde{\mathsf{EXT}}$ can be at most the total number of messages, which is $M = M(n)$.

Assume the claim is true for $d = d' + 1$. We show that it holds also for $d = d'$. The procedure $\widetilde{\mathsf{EXT}}$ simulates an execution with A in a straight-line on recursion level d', until it encounters the completion of a right-row s that has a $d' + 1$-good $\mathsf{safe\text{-}point}$ ρ, then it tries to obtain another transcript of s, by repeatedly invoking $\widetilde{\mathsf{EXT}}$ on recursion level $d' + 1$ from (the partial transcript) ρ. Hence, the number of queries made by $\widetilde{\mathsf{EXT}}$ is bounded by the sum of the number of queries made on level d', and the queries made by the recursive calls: the former is at most the total number of messages, that is, M, while the latter is bounded by the sum of the queries made by those recursive calls invoked for every right-row s. Furthermore we compute the expected number of queries made by the recursive calls for a right-row s by taking expectation over all partial transcript that is potentially a d'-good $\mathsf{safe\text{-}point}$ for s. let Γ_i denote the set of all partial transcripts of length i that are consistent with $\mathcal{V}$; for every $\rho \in \Gamma_i$, we denote by $\Pr[\rho \text{ occurs on level } d']$ the probability that ρ occurs (in the simulation) on level d', and $E[Q^s_{d'}(\rho)|\rho]$ the expected number of queries made by the recursive calls started from ρ for the right-row s, conditioned on ρ occurring on level d'. Then

$$E[\text{number of queries by } \widetilde{\mathsf{EXT}}] = M + \sum_s \sum_i \sum_{\rho \in \Gamma_i} Pr[\rho \text{ occurs on level } d'] \; E[Q^s_{d'}(\rho)|\rho]$$

Next we bound $E[Q^s_{d'}(\rho)|\rho]$ in two steps: the first step bounds the expected number of recursive calls started from ρ for proof s, and the second step uses the induction hypothesis to derive a bound on $E[Q^s_{d'}(\rho)|\rho]$.

Step 1: Given a partial transcript ρ from Γ_i, let $p^s_{d'}(\rho)$ denote the probability that conditioned on ρ occurring on level d', $\widetilde{\mathsf{EXT}}$ starts recursive calls from ρ for the right-row s, which happens if and only if ρ is a $d' + 1$-good $\mathsf{safe\text{-}point}$ for proof s, that is,

$$p^s_{d'}(\rho) = \Pr\left[\ \rho \text{ is } d' + 1\text{-good at level } d' \mid \rho\right]$$

When this happens, $\widetilde{\mathsf{EXT}}$ repeatedly calls itself on recursion level $d'+1$, until an invocation succeeds without cancelling. Let $q^s_{d'}(\rho)$ denote the probability that conditioned on ρ occurring on level d', a recursive call to $\widetilde{\mathsf{EXT}}$ on level $d'+1$ succeeds without cancelling. Since an invocation is cancelled if and only if ρ fails to be a $d'+1$-good $\mathsf{safe\text{-}point}$ for s in the invocation on level $d'+1$, we have

$$q^s_{d'}(\rho) = \Pr\left[\ \rho \text{ is } d'+1\text{-good at level } d'+1 \mid \rho\right]$$

We claim that $q^s_{d'}(\rho) \geq p^s_{d'}(\rho)$. This is because, conditioned on ρ occurring, the view of A on levels d' and $d'+1$ are simulated identically: on both levels d' and $d'+1$, $\widetilde{\mathsf{EXT}}$ emulates messages in the commitments of $\langle C, R\rangle$ for A perfectly; and furthermore, whenever A expects a committed value of a right interaction, $\widetilde{\mathsf{EXT}}$ sends it the value returned by the oracle $\mathcal{O}^*$, which is deterministic; thus A always receives the same value on both level d' and $d'+1$.

Then conditioned on ρ occurring on level d', the expected number of recursive invocations to level $d'+1$ before encountering a successful one is $1/q^s_{d'}(\rho)$. Since $\widetilde{\mathsf{EXT}}$ only starts recursive invocations from ρ with probability $p^s_{d'}(\rho)$, we have that the expected number of recursive calls from ρ for proof s, conditioned on ρ occurring on level d', is at most $p^s_{d'}(\rho)/q^s_{d'}(\rho) \leq 1$.

Step 2: From the induction hypothesis, we know that the expected number of queries made by an invocation of $\widetilde{\mathsf{EXT}}$ on level $d'+1$ is at most $\theta(d'+1)$. Therefore, if u recursive invocations are made from ρ for a right row s, the expected number of queries made is bounded by $u\theta(d'+1)$. Then we bound $E[Q^s_{d'}(\rho)|\rho]$ as follow:

$$\begin{aligned}
E[Q^s_{d'}(\rho)|\rho] &\leq \sum_{u\in N} \Pr\left[u \text{ recursive calls are made from } \rho \text{ for } s\right]\ u\ \theta(d'+1) \\
&= \theta(d'+1)\ \sum_{u\in N} \Pr\left[u \text{ recursive calls are made from } \rho \text{ for } s\right]\ u \\
&\leq \theta(d'+1)
\end{aligned}$$

Therefore,

$$\begin{aligned}
E[\text{number of queries by } \widetilde{\mathsf{EXT}}] &\leq M + \sum_s \sum_i \sum_{\rho\in\Gamma_i} \Pr\left[\rho \text{ occurs on level } d'\right]\ \theta(d'+1) \\
&= M + \theta(d'+1)\ \sum_s \sum_i \sum_{\rho\in\Gamma_i} \Pr\left[\rho \text{ occurs on level } d'\right] \\
&= M + \theta(d'+1)M^2 \\
&\leq M^{3(D-d'+1)} = \theta(d')
\end{aligned}$$

□

Combining Subclaim 1 and 2, we conclude that the expected running time of machine $\tilde{B}$ is bounded by a polynomial $t'(n)$. This concludes Claim 1.

Proof of Claim 2—Correctness of the Output distribution of $\tilde{B}$. We show that for every $b \in \{0,1\}$, the output of $\tilde{B}$ in $\mathsf{STA}_b(\langle\hat{C},\hat{R}\rangle, \tilde{B}^{\tilde{\mathcal{O}}}, n, z)$ is statistically close to $\mathsf{IND}_b(\langle C, R\rangle, A, n, z)$. Towards this, we show that conditioned on that $\tilde{B}$ does not output fail in $\mathsf{STA}_b(\langle\hat{C},\hat{R}\rangle, \tilde{B}^{\tilde{\mathcal{O}}}, n, z)$, the output of $\tilde{B}$ is identically distributed to $\mathsf{IND}_b(\langle C, R\rangle, A, n, z)$. By construction, $\tilde{B}$ invokes the recursive helper procedure $\widetilde{\mathsf{EXT}}$ (at the top recursion level $d=0$) and outputs fail if $\widetilde{\mathsf{EXT}}$ runs for more than 2^n steps. Conditioned on $\tilde{B}$ not outputting fail, $\tilde{B}$ returns the output of A contained in the simulated view $\mathcal{V}_A$ returned by $\widetilde{\mathsf{EXT}}$. As in $\mathsf{IND}_b(\langle C, R\rangle, A, n, z)$, the output is replaced with $\perp$

if A copies the identity of the left interaction in any right interaction. Hence it suffices to show that in the case where A does not copy the identity of the left interaction, $\widetilde{\mathsf{EXT}}$ simulates the messages in the left and right interactions for A perfectly. By construction of $\widetilde{\mathsf{EXT}}$, all the messages belonging to the commitments of $\langle C, R\rangle$ (both on the left and right) are simulated perfectly; furthermore, $\widetilde{\mathsf{EXT}}$ emulates the committed values of the right commitments using the values returned by the oracle $\tilde{\mathcal{O}}$, which are identical to the values extracted by the committed-value oracle $\mathcal{O}$ of $\langle C, R\rangle$. Therefore, the simulated view of A output by $\widetilde{\mathsf{EXT}}$ is identically distributed to the real view of A in IND_b. Finally, by Claim 1, the probability that $\tilde{B}$ runs for more than 2^n steps is exponentially small. Therefore we conclude that $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, D^{\tilde{\mathcal{O}}}, n, z)$ is statistically close to $\mathsf{IND}_b(\langle C, R\rangle, A, n, z)$.

Proof of Claim 3—$\tilde{B}$ almost never outputs $\widetilde{\mathsf{fail}}$. Assume for contradiction that there exist a polynomial p, $b \in \{0,1\}$ and an infinitely number of $n \in N$ and $z \in \{0,1\}^*$ such that $\tilde{B}$ in experiment $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, \tilde{B}^{\tilde{\mathcal{O}}}, n, z)$ outputs $\widetilde{\mathsf{fail}}$ with probability $1/p(n)$. Fix a b, n, and z for which this holds. Then by Claim 1, the probability that $\tilde{B}$ runs for more than $T(n) = 2t(n)p(n)$ steps is no more than $1/2p(n)$, where $t(n)$ is the expected running time of $\tilde{B}$. Then consider another machine $\tilde{B}_T$ that proceeds identically to $\tilde{B}$ except that it cuts-off the execution after $T(n)$ steps. We have that $\tilde{B}_T$ takes a strict polynomial number $T(n)$ of steps and the probability that $\tilde{B}_T$ outputs $\widetilde{\mathsf{fail}}$ is at least $1/2p(n)$.

Now consider another machine B^*_T that proceeds identically to B^* except that it also cuts-off the execution after $T(n)$ steps. We claim that in the experiment STA_b, B^*_T outputs fail or reconstructs a value for a right interaction that is not the valid committed value with polynomial probability. Assume for contradiction that this is false, that is, except with negligible probability, B^*_T always succeeds in reconstructing a valid committed value whenever the adversary expects a committed value during the rewindings. Then except with negligible probability, the committed values emulated by B^*_T are identical to that emulated by $\tilde{B}_T$. Therefore, the simulated view of A in B^*_T is statistically close to that in $\tilde{B}_T$. This implies that except with negligible probability, $\tilde{B}_T$ also always succeeds in reconstructing a valid committed value whenever the adversary expects one. Thus $\hat{B}_T$ outputs $\widetilde{\mathsf{fail}}$ only with negligible probability, which contradicts with our hypothesis. Therefore with polynomial probability, B^*_T fails to extract a valid committed value for some right interaction during its execution.

Below we reach a contradiction by showing that the probability that B^*_T outputs fail (Subclaim 3) and the probability that B^*_T reconstructs a value that is not the value committed value are negligible.

Subclaim 3. *For every $b \in \{0,1\}$, $n \in N$ and $z \in \{0,1\}^*$, the probability that B^*_T outputs* fail *during the execution of $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*_T, n, z)$ is negligible.*

Subclaim 4. *For every $b \in \{0,1\}$, $n \in N$ and $z \in \{0,1\}^*$, the probability that there exists a right interaction that is accepting and has an identity different from that of the left interaction, for which B^*_T reconstruct a value that is not the valid committed value in $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*_T, n, z)$ is negligible.*

Proof of Subclaim 3. Consider a fixed $b \in \{0,1\}$. By construction, B^*_T outputs fail if and only if one of the following cases occurs (in which B^* outputs fail).

Case 1: None of the rows in the right interaction is rewound.

Case 2: Some row is rewound and a pair of accepting transcripts of that row is collected, but that pair of transcripts is not admissible.

Case 3: A pair of admissible transcripts is collected, but the REC construction invoked with them outputs err.

Below we analyze the probabilities that each of the above cases occurs. We show that all these events occur with only negligible probability. Therefore, overall the probability that B_T^* outputs $\widetilde{\mathsf{fail}}$ is negligible.

Analysis of Case 1: We show that Case 1 never happens. More precisely, for every accepting right interaction j with a different identity from the left interaction, one of its rows must be rewound. By Lemma 6, there exist a number of $\Omega(\eta(n))$ non-overlapping rows in the right interaction j that has a $\mathsf{safe\text{-}point}$. Recall that in B_T^* (more precisely, in $\widetilde{\mathsf{EXT}}$), a right interaction may be carried out at multiple different recursion levels (through recursive calls); and at level d, B_T^* rewinds every row in this interaction that has a $d+1$-good $\mathsf{safe\text{-}point}$. By Subclaim 1, the recursion depth is only a constant; hence there must be a level d, on which a number of $\Omega(\eta(n))$ non-overlapping rows with a $\mathsf{safe\text{-}point}$ start in interaction j. Since the total number of right-rows that start on level d is bounded by $k_d = M/\eta'(n)^d$ (otherwise, the simulation is cancelled) and $\eta'(n) = o(\eta(n))$, there must exist one right-row that has a $\mathsf{safe\text{-}point}$ ρ, such that there are less than $M/\eta'(n)^{d+1}$ right-rows starting in between ρ and the last message of the row. Therefore ρ is a $d+1$-good $\mathsf{safe\text{-}point}$ for this right-row, and will be rewound.

Analysis of Case 2: Recall that a transcript of one row consists of a polynomial number of parallel commitments using Stage 2 of $\mathsf{PTrapCom}$, each of which consists of a polynomial number of parallel commitments using ExtCom. For a pair of transcripts of one row to be admissible, it must hold that all the pairs of transcripts it contains for each commitment of ExtCom are admissible w.r.t. ExtCom, that is, the two n-bit challenges in that pair of transcripts are different. Therefore, a pair of transcripts of a row is admissible if and only if the two challenge messages it contains, which are two tuples of a polynomial number q of n-bit strings $\alpha = (\alpha_1, \ldots, \alpha_q),\ \beta = (\beta_1, \ldots, \beta_q)$, are different at every location, that is, $\alpha_i \neq \beta_i$ for every $i \in [q]$.

We bound the probability that any n-bit challenge message is picked twice in whole execution of B_T^* to be negligible. Then since conditioned on this not happening, every two accepting transcripts of a row are admissible, we conclude that this case happens with negligible probability. Since B_T^* runs for at most $T(n)$ steps, it picks at most $T(n)$ n-bit challenges during the whole execution. By applying the union bound, we obtain that, the probability that a challenge β is picked again is at most $\frac{T(n)}{2^n}$, and hence, using the union bound again, the probability that *any* challenge in the execution is picked twice is at most $T(n)\frac{T(n)}{2^n}$. Hence, overall, the probability that this case occurs is negligible.

Analysis of Case 3: Let $\mathcal{T}_1$, $\mathcal{T}_2$ be a pair of admissible transcripts of one row of an accepting commitment $\mathcal{T}$ of $\langle C, R\rangle$. The REC procedure, on input $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}$, outputs err if and only if one of the invocations to the $\overline{\mathsf{reconst}}$ procedure returns err. Then by Lemma 4, the receiver's challenge in $\mathcal{T}$, which is shared among all the $\mathsf{TrapCom}$ commitments in $\mathcal{T}$, can be computed efficiently and deterministically from $\mathcal{T}_1$ and $\mathcal{T}_2$.

Then, suppose for contradiction that, there exists a polynomial g, such that for infinitely many $n \in N$ and z, case 3 occurs with probability at least $1/g(n)$ during the execution of B_T^*. Then the probability that case 3 occurs in a randomly chosen right interaction in B_T^* is at least $1/g(n)T(n)$. In other words, with polynomial probability, for a randomly chosen right interaction in B_T^*, REC is invoked and outputs err; then by the argument above, the receiver's challenge in this randomly chosen right interaction can be computed efficiently from the pair of admissible transcripts collected for this interaction (as input to REC). Furthermore, we note that in B_T^*, a pair of admissible transcripts is collected (if at all) for a right interaction,

before Stage 3 of that right interaction starts, and thus before the com commitment to the receiver's challenge is opened. Therefore we can use B^* to violate the hiding property of com.

More precisely, we construct a machine A^* that violates the hiding property of com. A^* on input a com commitment c to a random n-bit string e, internally emulates an interaction between $\hat{C}$ and B_T^*, except that it picks a random right interaction (in simulation by B_T^*) and feeds c as the Stage 1 message of that interaction; furthermore, after a pair of admissible transcripts $\mathcal{T}_1, \mathcal{T}_2$ is collected for this right interaction, A^* computes a challenge e' as described above; then, it aborts and outputs e'. Since A^* emulates an interaction between $\hat{C}$ and B_T^* perfectly before it aborts, the probability case 3 happens in a randomly chosen right interaction in A^* is identical to that in a randomly chosen right interaction in B_T^*, which is $1/g(n)T(n)$. Thus with polynomial probability, the challenge computed by A^* is the real challenge committed to in c. Thus A^* violates the hiding property of com.

□

Proof of Subclaim 4. Consider a fixed $b \in \{0,1\}$, n, z and a fixed right interaction $j \in [T(n)]$, we show that the probability that B_T^* reconstructs a value for the j^{th} right interaction that is not the valid committed value is negligible. Then it follows from a union bound that the probability that B_T^* reconstructs a value that is not the valid committed value in any right interaction is also negligible. If a value is reconstructed successfully for the j^{th} right interaction $\mathcal{T}$, it must be the case that interaction j is accepting, and a pair of admissible transcripts $\mathcal{T}_1, \mathcal{T}_2$ is collected and $\mathsf{REC}(\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}) = x \neq \mathsf{err}$ in $\mathcal{B}_T^*$. Then we show that except with negligible probability, x must be the committed value in $\mathcal{T}$.

Each row of a commitment of $\langle C, R\rangle$ contains $10n$ commitments of TrapCom. It follows from the strong computational binding property (see Lemma 5) of TrapCom that the probability that any of the TrapCom commitments generated by machine B_T^* has two valid committed value is negligible.[6] Therefore, the shares committed to using TrapCom in the j^{th} right interaction $\mathcal{T}$ are uniquely defined; let $s_1^i, \cdots, s_{10n}^i$ be the committed shares in the i^{th} row of $\mathcal{T}$ (s_k^i is set to $\perp$ if the k^{th} commitment in the i^{th} row is invalid). We say a column k is inconsistent if it contains a s_i^k equals to $\perp$ or two $s_{i_1}^k, s_{i_2}^k$ that are different; we claim that the probability that there are more than n inconsistent columns in the j^{th} right interaction is negligible. Recall that in the cut-and-choose stage of the right interaction j, B_T^* emulates the honest receiver's message by sending a randomly chosen subset Γ of size n; since the j^{th} right interaction is accepting, the adversary A must successfully reveal all the commitments in each column in Γ to the same value; therefore all the columns in Γ are consistent. Since Γ is chosen at random, the probability that the j^{th} right interaction contains more than n inconsistent columns, but none of them is chosen in Γ is exponentially small. Therefore, except with negligible probability, at least 0.9 fraction of the columns are consistent.

Now we are ready to show that the value x returned by the procedure REC on input a pair of admissible transcripts $\mathcal{T}_1, \mathcal{T}_2$ of the k^{th} row of the right commitment $\mathcal{T}$ is the valid committed value. From $\mathcal{T}_1, \mathcal{T}_2$, the procedure RECextracts $10n$ values $\tilde{s}_1^k, \cdots, \tilde{s}_{10n}^k$ corresponding to the $10n$ shares committed to in the k^{th} row of $\mathcal{T}$. Then consider the following two possible cases:

In the first case, the shares $s_1^1, \cdots, s_{10n}^1$ committed to in the first row of $\mathcal{T}$ is 0.9-close to a valid code w. Since except with negligible probability, at least 0.9 fraction of the columns are consistent, by the special soundness of TrapCom, the extracted shares $\tilde{s}_1^k, \cdots, \tilde{s}_{10n}^k$ and with $s_1^1, \cdots, s_{10n}^1$ agree with each other at no less than 0.9 fraction of positions. Therefore

[6] This also relies on the fact that TrapCom is a generalized public-coin protocol, in the sense, that given a partial transcript of a commitment of TrapCom, messages from the honest receiver continuing from that partial transcript can be emulated efficiently, by simply sending random strings.

$\tilde{s}_1^k, \cdots, \tilde{s}_{10n}^k$ is 0.8-close to w. Therefore the procedure REC will recover w uniquely. Then if w agrees with all the shares opened in the cut-and-choose stage in $\mathcal{T}$, the valid committed value is the value v encoded in w; in this case, REC also performs the same check and will output v as the committed value correctly. On the other hand, if w disagrees with one of the shares opened in the cut-and-choose stage in $\mathcal{T}$, the commitment is invalid and the committed value is set to $\perp$; in this case REC performing the same check, will also return $\perp$ correctly.

In the second case, the shares $s_1^1, \cdots, s_{10n}^1$ committed to in the first row of $\mathcal{T}$ is 0.1-far away from every valid code w. In this case, the commitment $\mathcal{T}$ is invalid and the committed value is set to $\perp$. We show that in this case, the probability that the procedure REC does not output $\perp$ is negligible. If REC outputs a value $v' \neq \perp$, it must be the case that $\tilde{s}_1^k, \cdots, \tilde{s}_{10n}^k$ is 0.8-close to a valid codeword w' that encodes v'. By our hypothesis, $s_1^1, \cdots, s_{10n}^1$ is 0.1-far away from w'. Since $\mathcal{T}$ is accepting, all the columns in Γ are consistent, and thus the shares revealed in the cut-and-choose stage equals to $\left\{s_i^1\right\}_{i\in\Gamma} \neq \perp$. By construction of REC, it outputs v' only if w' agrees with all the shares revealed in the cut-and-choose stage, that is, $s_i^1 = w_i'$ for every $i \in \Gamma$. However, since the set Γ is chosen at random by the honest receiver (emulated by B_T^*), the probability that w' disagrees with $s_1^1, \cdots, s_{10n}^1$ at more than n locations but none of them is selected in Γ is exponentially small. Therefore, except with negligible probability, REC outputs $\perp$ correctly.

Combining the above two cases, we conclude that, except with negligible probability, the values reconstructed by REC must be the valid committed value. □

B.3.3 Proof of Robustness

In this section, we extend the proof in the last section to show that $\langle C, R\rangle$ is also *robust* w.r.t. the committed-value oracle $\mathcal{O}$. Towards this, we need to show that for every $k \leq \kappa(n)$, and every $\mathcal{PPT}$ adversary A, there exists a simulator S, such that, for every $\mathcal{PPT}$ k-round ITM B, the interaction between B and A with access to $\mathcal{O}$ is indistinguishable from that between B and S. The construction of the simulator is similar to that in [CLP10], but the correctness follows from a proof similar to the proof of CCA in the last section. For completeness, we provide the construction of S below; but omit the proof.

Given an adversary A, and a constant k, the construction of the simulator S is very similar to that of B^* in the last section. On a high-level, S externally interacts with an arbitrary k-round ITM B, and internally simulates an execution between B and $A^{\mathcal{O}}$, by forwarding messages from B internally to A, while concurrently extracting the committed values of the right interactions from A to simulate $\mathcal{O}$. The extraction strategy of S is essentially the same as that used by B^*: it recursively rewinds A over the rows in Stage 2 of the protocol to extract the committed values, except that, here the goal is to make sure that the left interaction with B is *never* rewound, (instead of the goal of ensuring that the left interaction remains hiding (in B^*)). This is achieved by rewinding only those right-rows that do not interleave with any messages in the left interaction, and cancelling every rewinding in which the right-row interleaves with a left-message. More precisely, consider the notion of R-safe-point (which is in analogous to the notion of safe-point)—a prefix ρ of a transcript Δ is a R-safe-point for a right-row (α, β, γ) if it includes all the messages in Δ up to α (inclusive), and that no left-message is exchanged in between ρ and γ. Then S simply runs the procedure EXTdefined in the last section internally, except that it replaces the notion of safe-point with R-safe-point, and that it simulates the left interaction with A by forwarding the messages between A and B; everything else remains the same. Then it follows from the fact that S always rewinds A from a R-safe-point ρ, and cancels every rewindings in which ρ is not a R-safe-point, the left interaction is never rewound. Furthermore, since the left interaction with B consists of only $k \leq \kappa(n)$ rounds, and the protocol $\langle C, R\rangle$ contains at least $\kappa(n) + \eta(n)$ rows, there exist at least $\eta = n^{\varepsilon}$ R-safe-point in every successful

right interaction. Then, it follows from the same proof as in Claim 7 and Claim 8 that there is a polynomial t, such that the probability that S takes more than $T(n)$ steps is smaller than $t(n)/T(n)$ plus a negligible amount, and that the joint output of S and B is indistinguishable from that of $A^{\mathcal{O}}$ and B.

C Model of Security

C.1 UC and Global UC security

We briefly review UC and externalized UC (EUC) security. For full details see [Can00, CDPW07]. The original motivation to define EUC security was to capture settings where all ITMs in the system have access to some global, potentially trusted information (such as a globally available public key infrastructure or a bulletin board) [CDPW07]. Here however we use the EUC formalism to capture the notion of global helper functionalities that are available only to the corrupted parties.

We first review the model of computation, ideal protocols, and the general definition of securely realizing an ideal functionality. Next we present hybrid protocols and the composition theorem.

The basic model of execution. Following [GMR89, Gol01], a protocol is represented as an interactive Turing machine (ITM), which represents the program to be run within each participant. Specifically, an ITM has three tapes that can be written to by other ITMs: the input and subroutine output tapes model the inputs from and the outputs to other programs running within the same "entity" (say, the same physical computer), and the incoming communication tapes and outgoing communication tapes model messages received from and to be sent to the network. It also has an identity tape that cannot be written to by the ITM itself. The identity tape contains the program of the ITM (in some standard encoding) plus additional identifying information specified below. Adversarial entities are also modeled as ITMs.

We distinguish between ITMs (which represent static objects, or programs) and *instances of ITMs*, or *ITI*s, that represent interacting processes in a running system. Specifically, an ITI is an ITM along with an identifier that distinguishes it from other ITIs in the same system. The identifier consists of two parts: A session-identifier (SID) which identifies which protocol instance the ITI belongs to, and a party identifier (PID) that distinguishes among the parties in a protocol instance. Typically the PID is also used to associate ITIs with "parties", or clusters, that represent some administrative domains or physical computers.

The model of computation consists of a number of ITIs that can write on each other's tapes in certain ways (specified in the model). The pair (SID,PID) is a unique identifier of the ITI in the system. With one exception (discussed within) we assume that all ITMs are probabilistic polynomial time.[7]

Security of protocols. Protocols that securely carry out a given task (or, protocol problem) are defined in three steps, as follows. First, the process of executing a protocol in an adversarial environment is formalized. Next, an "ideal process" for carrying out the task at hand is formalized. In the ideal process the parties do not communicate with each other. Instead they have access to an "ideal functionality," which is essentially an incorruptible "trusted party" that is programmed to capture the desired functionality of the task at hand. A protocol is said to securely realize an ideal functionality if the process of running the protocol amounts to "emulating" the ideal process

[7] An ITM is $\mathcal{PPT}$ if there exists a constant $c > 0$ such that, at any point during its run, the overall number of steps taken by M is at most n^c, where n is the overall number of bits written on the *input tape* of M in this run. In fact, in order to guarantee that the overall protocol execution process is bounded by a polynomial, we define n as the total number of bits written to the input tape of M, *minus the overall number of bits written by M to input tapes of other ITMs*; see [Can01].

for that ideal functionality. Below we overview the model of protocol execution (called the *real-life model*), the ideal process, and the notion of protocol emulation.

The model for protocol execution. The model of computation consists of the parties running an instance of a protocol π, an adversary A that controls the communication among the parties, and an environment Z that controls the inputs to the parties and sees their outputs. We assume that all parties have a security parameter $k \in \mathbf{N}$. (We remark that this is done merely for convenience and is not essential for the model to make sense). The execution consists of a sequence of *activations,* where in each activation a single participant (either Z, A, or some other ITM) is activated, and may write on a tape of at most *one* other participant, subject to the rules below. Once the activation of a participant is complete (i.e., once it enters a special waiting state), the participant whose tape was written on is activated next. (If no such party exists then the environment is activated next.)

The environment is given an external input z and is the first to be activated. In its first activation, the environment invokes the adversary A, providing it with some arbitrary input. In the context of UC security, the environment can from now on invoke (namely, provide input to) only ITMs that consist of a single instance of protocol π. That is, all the ITMs invoked by the environment must have the same SID and the code of π. In the context of EUC security the environment can in addition invoke an additional ITI that interacts with all parties. We call this ITI the helper functionality, denoted $\mathcal{H}$.

Once the adversary is activated, it may read its own tapes and the outgoing communication tapes of all parties. It may either deliver a message to some party by writing this message on the party's incoming communication tape or report information to Z by writing this information on the subroutine output tape of Z. For simplicity of exposition, in the rest of this paper we assume authenticated communication; that is, the adversary may deliver only messages that were actually sent. (This is however not essential since authentication can be realized via a protocol, given standard authentication infrastructure [Can04].)

Once a protocol party (i.e., an ITI running π) is activated, either due to an input given by the environment or due to a message delivered by the adversary, it follows its code and possibly writes a local output on the subroutine output tape of the environment, or an outgoing message on the adversary's incoming communication tape.

The protocol execution ends when the environment halts. The output of the protocol execution is the output of the environment. Without loss of generality we assume that this output consists of only a single bit.

Let $\text{EXEC}_{\pi,A,Z}(k,z,r)$ denote the output of the environment Z when interacting with parties running protocol π on security parameter k, input z and random input $r = r_Z, r_A, r_1, r_2, ...$ as described above (z and r_Z for Z; r_A for A, r_i for party P_i). Let $\text{EXEC}_{\pi,A,Z}(k,z)$ denote the random variable describing $\text{EXEC}_{\pi,A,Z}(k,z,r)$ when r is uniformly chosen. Let $\text{EXEC}_{\pi,A,Z}$ denote the ensemble $\{\text{EXEC}_{\pi,A,Z}(k,z)\}_{k\in N, z\in\{0,1\}^*}$.

Ideal functionalities and ideal protocols. Security of protocols is defined via comparing the protocol execution to an *ideal protocol* for carrying out the task at hand. A key ingredient in the ideal protocol is the *ideal functionality* that captures the desired functionality, or the specification, of that task. The ideal functionality is modeled as another ITM (representing a "trusted party") that interacts with the parties and the adversary. More specifically, in the ideal protocol for functionality $\mathcal{F}$ all parties simply hand their inputs to an ITI running $\mathcal{F}$. (We will simply call this ITI $\mathcal{F}$. The SID of $\mathcal{F}$ is the same as the SID of the ITIs running the ideal protocol. (the PID of $\mathcal{F}$ is null.)) In addition, $\mathcal{F}$ can interact with the adversary according to its code. Whenever $\mathcal{F}$ outputs a value to a party, the party immediately copies this value to its own output tape. We call the parties in the ideal protocol dummy parties. Let $\pi(\mathcal{F})$ denote the ideal protocol for functionality $\mathcal{F}$.

Securely realizing an ideal functionality. We say that a protocol π *emulates* protocol ϕ if for any adversary A there exists an adversary $\mathcal{S}$ such that no environment Z, on any input, can tell with non-negligible probability whether it is interacting with A and parties running π, or it is interacting with $\mathcal{S}$ and parties running ϕ. This means that, from the point of view of the environment, running protocol π is 'just as good' as interacting with ϕ. We say that π *securely realizes* an ideal functionality $\mathcal{F}$ if it emulates the ideal protocol $\pi(\mathcal{F})$. More precise definitions follow. A distribution ensemble is called *binary* if it consists of distributions over $\{0,1\}$.

Definition 9. *Let π and ϕ be protocols. We say that π* UC-emulates *(resp.,* EUC-emulates*) ϕ if for any adversary A there exists an adversary $\mathcal{S}$ such that for any environment Z that obeys the rules of interaction for UC (resp., EUC) security we have* $\text{EXEC}_{\phi,\mathcal{S},Z} \approx \text{EXEC}_{\pi,A,Z}$.

Definition 10. *Let $\mathcal{F}$ be an ideal functionality and let π be a protocol. We say that π* UC-realizes *(resp.,* EUC-realizes*) $\mathcal{F}$ if π* UC-emulates *(resp.,* EUC-emulates*) the ideal protocol $\pi(\mathcal{F})$.*

Security with dummy adversaries. Consider the adversary $\mathcal{D}$ that simply follows the instructions of the environment. That is, any message coming from one of the ITIs running the protocol is forwarded to the environment, and any input coming from the environment is interpreted as a message to be delivered to the ITI specified in the input. We call this adversary the dummy adversary. A convenient lemma is that UC security with respect to the dummy adversary is equivalent to standard UC security. That is:

Definition 11. *Let π and ϕ be protocols. We say that π* UC-emulates *(resp.,* EUC-emulates*) ϕ* w.r.t the dummy adversary *$\mathcal{D}$ if there exists an adversary $\mathcal{S}$ such that for any environment Z that obeys the rules of interaction for UC (resp., EUC) security we have* $\text{EXEC}_{\phi,\mathcal{S},Z} \approx \text{EXEC}_{\pi,\mathcal{D},Z}$.

Theorem 3. *Let π and ϕ be protocols. Then π* UC-emulates *(resp.,* EUC-emulates*) ϕ if and only if π* UC-emulates *(resp.,* EUC-emulates*) ϕ* with respect to the dummy adversary.

Hybrid protocols. Hybrid protocols are protocols where, in addition to communicating as usual as in the standard model of execution, the parties also have access to (multiple copies of) an ideal functionality. Hybrid protocols represent protocols that use idealizations of underlying primitives, or alternatively make *trust assumptions* on the underlying network. They are also instrumental in stating the universal composition theorem. Specifically, in an $\mathcal{F}$-hybrid protocol (i.e., in a hybrid protocol with access to an ideal functionality $\mathcal{F}$), the parties may give inputs to and receive outputs from an unbounded number of copies of $\mathcal{F}$.

The communication between the parties and each one of the copies of $\mathcal{F}$ mimics the ideal process. That is, giving input to a copy of $\mathcal{F}$ is done by writing the input value on the input tape of that copy. Similarly, each copy of $\mathcal{F}$ writes the output values to the subroutine output tape of the corresponding party. It is stressed that the adversary does not see the interaction between the copies of $\mathcal{F}$ and the honest parties.

The copies of $\mathcal{F}$ are differentiated using their SIDs. All inputs to each copy and all outputs from each copy carry the corresponding SID. The model does not specify how the SIDs are generated, nor does it specify how parties "agree" on the SID of a certain protocol copy that is to be run by them. These tasks are left to the protocol. This convention seems to simplify formulating ideal functionalities, and designing protocols that securely realize them, by freeing the functionality from the need to choose the SIDs and guarantee their uniqueness. In addition, it seems to reflect common practice of protocol design in existing networks.

The definition of a protocol securely realizing an ideal functionality is extended to hybrid protocols in the natural way.

The universal composition operation. We define the universal composition operation and state the universal composition theorem. Let ρ be an $\mathcal{F}$-hybrid protocol, and let π be a protocol that securely realizes $\mathcal{F}$. The composed protocol ρ^π is constructed by modifying the code of each ITM in ρ so that the first message sent to each copy of $\mathcal{F}$ is replaced with an invocation of a new copy of π with fresh random input, with the same SID, and with the contents of that message as input. Each subsequent message to that copy of $\mathcal{F}$ is replaced with an activation of the corresponding copy of π, with the contents of that message given to π as new input. Each output value generated by a copy of π is treated as a message received from the corresponding copy of $\mathcal{F}$. The copy of π will start sending and receiving messages as specified in its code. Notice that if π is a $\mathcal{G}$-hybrid protocol (i.e., ρ uses ideal evaluation calls to some functionality $\mathcal{G}$) then so is ρ^π.

The universal composition theorem. Let $\mathcal{F}$ be an ideal functionality. In its general form, the composition theorem basically says that if π is a protocol that UC-realizes $\mathcal{F}$ (resp., *EUC*-realizes $\mathcal{F}$) then, for any $\mathcal{F}$-hybrid protocol ρ, we have that an execution of the composed protocol ρ^π "emulates" an execution of protocol ρ. That is, for any adversary A there exists a simulator $\mathcal{S}$ such that no environment machine Z can tell with non-negligible probability whether it is interacting with A and protocol ρ^π or with $\mathcal{S}$ and protocol ρ, in a UC (resp., EUC) interaction. As a corollary, we get that if protocol ρ UC-realizes $\mathcal{F}$ (resp., EUC-realizes $\mathcal{F}$), then so does protocol ρ^π.[8]

Theorem 4 (Universal Composition [Can01, CDPW07])**.** *Let $\mathcal{F}$ be an ideal functionality. Let ρ be a $\mathcal{F}$-hybrid protocol, and let π be a protocol that UC-realizes $\mathcal{F}$ (resp., EUC-realizes $\mathcal{F}$). Then protocol ρ^π UC-emulates ρ (resp., EUC-emulates ρ).*

An immediate corollary of this theorem is that if the protocol ρ UC-realizes (resp., EUC-realizes) some functionality $\mathcal{G}$, then so does ρ^π.

C.2 UC Security with Super-polynomial Helpers

We modify the definitions of UC security by giving the corrupted parties access to an external "helper" entity, in a conceptually similar way to [PS04]. This entity, denoted $\mathcal{H}$, is computationally unbounded, and can be thought of as providing the corrupted parties with some judicious help. (As we'll see, this help will be used to assist the simulator to "reverse engineering" the adversary in order to extract relevant information hidden in its communication.)

The definition uses the formalism of EUC security [CDPW07]. Specifically, we model the helper entity as an ITM that is invoked directly by the environment, and that interacts with the environment and the corrupted parties. More formally, let $\mathcal{H}$ be an ITM. An environment Z is called *aided by* $\mathcal{H}$ if: (a) Z invokes a single instance $\mathcal{H}$ immediately after invoking the adversary; (b) As soon as a party (i.e., an ITI) P is corrupted (i.e., P receives a `corrupted` message), Z lets $\mathcal{H}$ know of this fact; (c) $\mathcal{H}$ interacts only with the corrupted parties. Then:

Definition 12. *Let π and ϕ be protocols, and let $\mathcal{H}$ be a helper functionality (i.e., an ITM). We say that π $\mathcal{H}$-*EUC*-emulates ϕ if for any adversary A there exists an adversary $\mathcal{S}$ such that for any environment Z that's aided by $\mathcal{H}$ we have $\text{EXEC}_{\phi,\mathcal{S},Z} \approx \text{EXEC}_{\pi,A,Z}$.*

The meaningfulness of relativized UC security of course depends on the particular helper ITM in use. Still, it is easy to see that if protocol π $\mathcal{H}$-EUC-emulates protocol ϕ where $\mathcal{H}$ obeys the above rules and runs in time $T(n)$, then π UC-emulates ϕ according to a relaxed notion where the adversary $\mathcal{S}$ can run in time $poly(T(n))$. As noted in the past, for many protocols and ideal

[8] The universal composition theorem in [Can01] applies only to "subroutine respecting protocols", namely protocols that do not share subroutines with any other protocol in the system. In [CDPW07] the theorem is extended to protocols that share subroutines with arbitrary other protocols, as long as the composed protocol, ρ^π, realizes $\mathcal{F}$ with EUC security.

functionalities, this relaxed notion of security suffices even when $T(n) = exp(n)$ [Pas03b, PS04, BS05, MMY06].

Universal Composition with super-polynomial helpers. The universal composition theorem generalizes naturally to the case of EUC, even with super-polynomial helper functionalities:

Theorem (universal composition for relativized UC)**.** *Let $\mathcal{F}$ be an ideal functionality, let $\mathcal{H}$ be a helper functionality, let π be an $\mathcal{F}$-hybrid protocol, and let ρ be a protocol that $\mathcal{H}$-EUC-realizes $\mathcal{F}$. Then protocol π^ρ $\mathcal{H}$-EUC-emulates π.*

Proof. The proof of Theorem C.2 follows the same steps as the proof of Theorem 4 (see e.g. the proof in [Can00]). The only difference is in the construction of the distinguishing environment Z_π (see there). Recall that Z_π takes an environment Z that distinguishes between an execution of π and an execution of π^ρ, and uses it to distinguish between an execution of ρ and an ideal evaluation of $\mathcal{F}$. For this purpose, Z_π emulates for Z an execution of π^ρ.

Now, in the presence of the helper $\mathcal{H}$, Z_ρ must emulate for Z also the interaction with $\mathcal{H}$. Note that Z_π cannot run $\mathcal{H}$ on its own, since $\mathcal{H}$ may well be super-polynomial in complexity. Instead, Z_π will forward to the external instance of $\mathcal{H}$ each message sent to $\mathcal{H}$ by Z. Similarly, whenever any of the corrupted parties that Z_π locally runs sends a message to $\mathcal{H}$, Z_π externally invokes a party with the same ID and code, corrupts it, and instructs it to send the query to the external instance of $\mathcal{H}$. The responses of $\mathcal{H}$ are handled analogously.

Note that the proof uses the fact that the helper functionality $\mathcal{H}$ does not take messages directly from the adversary. Indeed, Z_π cannot emulate for the external instance of $\mathcal{H}$ messages coming from the adversary. □

D Black-Box UC-Secure Protocols in $\mathcal{H}$-EUC Model

In this work, we consider UC-security with the a super-polynomial time helper that help breaks commitments of our black-box robust CCA secure commitment scheme $\langle C, R\rangle$. More precisely, it proceeds as described in Figure 6

Functionality $\mathcal{H}$

Corrupted Parties: Upon receiving an input (`Corrupt`, P_i, sid) from the environment, record (`Corrupt`, P_i, sid).

Initialization: Upon receiving an input (`Init`,P_i, sid, k) from party P_i in the protocol instance sid, if there is no previously recorded tuple (`Corrupt`, P_i, sid) or there is a previously recorded session (P_i, sid, k), ignore this message; otherwise, initialize a session of $\langle C, R\rangle$ with $\mathcal{O}$ using identity (P_i, sid), and record session (P_i, sid, k).

Accessing $\mathcal{O}$: Upon receiving an input (`Mesg`,P_i, sid, k, m) from party P_i in the protocol instance sid, if there is no previously recorded session (P_i, sid, k), ignore the message; otherwise, forward m to $\mathcal{O}$ in the k^{th} session that uses identity (P_i, sid), obtain a reply m', and return (`Mesg`,P_i, sid, k, m') to P_i.

Figure 6: The ideal functionality $\mathcal{H}$

D.1 Proof of Lemma 2

In this section, we first recall the protocol Π_{OT} that $\mathcal{H}$-EUC emulates $\mathcal{F}_{OT}$ and then prove its security. The construction relies on the T_{OT}-round mS-OT protocol $\langle S, R\rangle$ and the CCA-secure

commitment scheme $\langle C, R\rangle$. On common input 1^n, the sender and the receiver of the protocol Π_{OT} on private inputs (v_0, v_1) and u respectively proceed as follow:

Stage 1: The sender chooses a random subset $\Gamma_R \subseteq [20n]$ of size n and commits to Γ_R using $\langle C, R\rangle$.

The receiver chooses a random subset $\Gamma_S \subseteq [20n]$ of size n and another random subset $\Gamma \subseteq [18n]$ of size n; it then commits to both Γ_S and Γ using $\langle C, R\rangle$.

Stage 2 (Coin-Tossing):

Receiver Random-Tape Generation: The receiver chooses $20n$ random strings $(a_1^R, \ldots a_{20n}^R)$ and commits to them using $\langle C, R\rangle$. The sender sends $20n$ random strings $(b_1^R, \ldots b_{20n}^R)$. The receiver calculates $r_i^R = a_i^R \oplus b_i^R$ for every $i \in [20n]$, and interprets r_i^R as $c_i \| \tau_i^R$, where c_i will be used as the receiver's input bit, and τ_i^R the random tape in the OT executions below.

Sender Random-Tape Generation: The sender chooses $20n$ random strings $(a_1^S, \ldots a_{20n}^S)$ and commits to them using $\langle C, R\rangle$. The receiver sends $20n$ random strings $(b_1^S, \ldots b_{20n}^S)$. The sender calculates $r_i^S = a_i^S \oplus b_i^S$ for every $i \in [20n]$, and interprets r_i^S as $s_i^0 \| s_i^1 \| \tau_i^S$, where s_i^0 and s_i^1 will be used as the sender's two input bits, and τ_i^S the random tape in the OT executions below.

Stage 3 (OT with Random Inputs): The sender and the receiver participates in $20n$ executions of the OT protocol $\langle S, R\rangle$ in parallel, where the sender acts as S and the receiver acts as R. In the i^{th} execution of $\langle S, R\rangle$, the sender uses inputs s_i^0, s_i^1 and random tape r_i^S and the receiver uses input c_i and random tape r_i^R. At the end of the execution, the receiver obtains outputs $\tilde{s}_1 \ldots \tilde{s}_{20n}$.

Stage 4 (Cut-and-Choose—Honesty Checking):

Sender Honesty Checking: The receiver opens Γ_S and sender responds as follows: for every $j \in \Gamma_S$, the sender opens the j^{th} commitments of $\langle C, R\rangle$ in Stage 2 to $\tilde{a}_j^S$. The receiver checks if the openings are valid, and if for every $j \in \Gamma_S$, the sender acted honestly in the j^{th} OT execution according to $\tilde{a}_j^S \oplus b_j^S$. The receiver aborts if not.

Receiver Honesty Checking: The sender opens Γ_R and receiver responds as follows: for every $j \in \Gamma_R$, the receiver opens the j^{th} commitments of $\langle C, R\rangle$ in Stage 2 to $\tilde{a}_j^R$. The sender checks if the openings are valid and if for every $j \in \Gamma_R$, the receiver acted honestly in the j^{th} OT execution according to $\tilde{a}_j^R \oplus b_j^R$. The sender aborts if not.

Stage 5 (Combiner): Set $\Delta = [20n] - \Gamma_R - \Gamma_S$ (i.e., Δ is the set of unopened locations). For every $i \in \Delta$ The receiver computes $\alpha_i = u \oplus c_i$ and sends α_i. The sender responds as follows: It computes a $10n$-out-of-$18n$ secret-sharing of v_0; without loss of generality, we index shares in that secret-sharing with elements in Δ; let the secret-sharing be $\rho^0 = \left\{\rho_i^0\right\}_{i\in\Delta}$. Similarly, it also computes a $10n$-out-of-$18n$ secret-sharing $\rho^1 = \left\{\rho_i^1\right\}_{i\in\Delta}$ for v_1. Then the sender computes $\beta_i^b = \rho_i^b \oplus s_i^{b\oplus\alpha_i}$ for every $i \in \Delta$ and sends back all the β_i^b's.

The receiver after receiving all the β_i^b's, computes shares corresponding to the u^{th} input as $\tilde{\rho}_i = \beta_i^u \oplus \tilde{s}_i$ for every $i \in \Delta$, and sets $\tilde{\rho} = \{\tilde{\rho}_i\}_{i\in\Delta}$.

Stage 6 (Cut-and-Choose—Consistency Checking): The receiver opens to Γ. Then for every $j \in \Gamma \cap \Delta$, the sender reveals the two inputs $\hat{s}_j^0$ and $\hat{s}_j^1$ and random tape $\hat{\tau}_j^S$ that it uses in the j^{th} OT execution in Stage 3. The receiver checks if the sender acts honestly according to input $(\hat{s}_j^0, \hat{s}_j^1)$ and random tape $\hat{\tau}_j^S$ and aborts if not.

Finally the receiver checks whether $\tilde{\rho}$ computed in Stage 5 is $17n$-close to a valid codeword w (that is, it agrees with w at $17n$ locations), and if for every $j \in \Gamma \cap \Delta$, w_j is equal to $\beta_j^u \oplus \hat{s}_j^{u \oplus \alpha_j}$. If so it outputs the value v encoded in w; otherwise, it aborts.

Next we proceed to show that Π_{OT} is indeed a secure realization of $\mathcal{F}_{\mathsf{OT}}$. Below we describe the technique for simulating the protocol execution of Π_{OT} in the ideal-world, where parties have access to the ideal commitment functionality $\mathcal{F}_{\mathsf{OT}}$, and give a proof that the simulation in the ideal-world setting is indistinguishable from a real-world execution of Π_{OT}. Recall that we only need to prove that Π_{OT} $\mathcal{H}$-EUC-emulates $\mathcal{F}_{\mathsf{OT}}$; hence in both the ideal and real worlds, the environment and the adversary have access to the $\mathcal{H}$ functionality.

Let A be any $\mathcal{PPT}$ adversary and Z any $\mathcal{PPT}$ environment. The simulator Sim for A in the ideal world internally simulates a real-world execution with A on auxiliary input z: it simulates A's interaction with the environment Z and the functionality $\mathcal{H}$, by simply forwarding the communications between A and Z or $\mathcal{H}$; furthermore, it simulates messages belonging to the OT protocol Π_{OT} for A as follows:

Strategy 1: If the Sender (P_i) is honest and the Receiver (P_j) is corrupted, the simulator needs to be able to extract the choice u of the receiver (controlled by A) so that it can send u to the ideal OT functionality $\mathcal{F}_{\mathsf{OT}}$ to obtain an input v_u, and simulate the view of A without knowing the other input v_{1-u}.

Towards this, the simulator Sim first acts honestly in Stage 1 to 4 except the following: It forwards all the commitments of $\langle C, R\rangle$ from A in Stage 1 and 2 to the helper functionality $\mathcal{H}$. Since the receiver P_j is corrupted, $\mathcal{H}$ accepts commitments with identity P_j from Sim, and returns Sim the unique committed value if the commitment is valid and $\perp$ otherwise. These committed values include Γ_S and Γ committed to by A in Stage 1 and all the random strings a_i^R for $i \in [20n]$ committed to by A in Stage 2, which allows Sim to recover the inputs and random tapes $\left\{c_i, \tau_i^R\right\}_{i\in[20n]}$ that A is supposed to use in the Stage 3 OT executions. Then for every $j \in [20n]$, Sim checks whether A behaves honestly according to c_j, τ_j^R in the j^{th} OT execution in Stage 3, and sets Φ to be the set of locations in which A cheats. Next, if A successfully completes the first 4 stages, Sim needs to extract its input choice u and simulate the Stage 5 sender's message. To do so, it first extracts u by counting how many shares out of the $18n$ shares $\rho^0 = \left\{\rho_i^0\right\}_{i\in\Delta}$ and $\rho^1 = \left\{\rho_i^1\right\}_{i\in\Delta}$ that A will get (in Stage 5 and 6) for each input v_0 and v_1 as follows:

- For every location $j \in \Delta$ and also in Γ, since the sender's inputs s_j^0 and s_j^1 will be revealed in stage 6, Sim counts that A obtains one more share for both ρ^0 and ρ^1.
- For every location $j \in \Delta$ and also in Φ, A has cheated in the j^{th} OT in Stage 3 and thus may obtain both of the sender's inputs s_j^0 and s_j^1 in that OT execution; recall that in Stage 5 of the protocol, the two shares ρ_j^0 and ρ_j^1 will be covered using the sender's inputs s_j^0 and s_j^1 as one-time pads. Therefore, after receiving the Stage 5 message (which contains $\beta_j^b = \rho_j^b \oplus s_j^{b\oplus\alpha_j}$), A will be able to recover both shares. Thus Sim counts that A again obtains one more share for both ρ^0 and ρ^1.
- For the rest of locations $j \in \Delta - \Phi - \Gamma$, since A acts honestly using input c_j and the two sender's inputs s_j^0 and s_j^1 will not be revealed in Stage 6, A obtains $s_j^{c_j}$ through the OT execution while $s_j^{1-c_j}$ remains computationally hidden. Thus A later can only recover the share $\rho_j^{c_j\oplus\alpha_j}$. Therefore, Sim counts that A gets one more share of $\rho^{c_j\oplus\alpha_j}$.

Then to simulates the sender's Stage 5 message, Sim proceeds as follows: If for both inputs A gets more than $10n$ shares, the simulator Sim outputs fail and aborts. Otherwise, if for only

one input A gets more than $10n$ shares, Sim sends its index b^* to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$ and receives a value w; it then sets $v_{b^*} = w$ and sets v_{1-b^*} to a random bit, and complete the rest of the simulation by following the honest sender's strategy using v_{b^*} and v_{1-b^*} as inputs. Finally, if for none of the inputs A gets more than $10n$ shares, then Sim simply sets both v_0 and v_1 to random bits and complete the simulation honestly according to these two values.

Strategy 2: If the Sender (P_i) is corrupted and the Receiver (P_j) is honest, the simulator needs to simulate the view of the sender (controlled by A) without knowing the choice of the receiver and extracts the two inputs from A.

Towards this, first note that during the whole execution of Π_{OT}, the only message that depends on the receiver's choice u is the Stage 5 receiver's message, consisting of $\{\alpha_j\}_{j\in\Delta}$ that are supposed to be set to $\alpha_j = u \oplus c_j$. The simulator Sim simulates the α_j's by simply sending random bits (and emulates the rest of the receiver's messages for A honestly). Furthermore, to extract the two inputs of the sender (controlled by A), Sim proceeds as follows: It forwards all the commitments of $\langle C, R\rangle$ from A in Stage 1 and 2 to the helper functionality $\mathcal{H}$. Since the sender P_i is corrupted, $\mathcal{H}$ accepts commitments with identity P_i from Sim, and returns Sim the unique committed value if the commitment is valid and $\perp$ otherwise. These committed values include Γ_R committed to by A in Stage 1 and all the random strings a_i^S for $i \in [20n]$ committed to by A in Stage 2, which allows Sim to recover the inputs and random tapes $\left\{s_i^0, s_i^0, \tau_i^R\right\}_{i\in[20n]}$ that A is supposed to use (as a sender) in the Stage 3 OT executions. Next, if A completes the execution of Π_{OT} successfully, Sim extracts shares of the sender's inputs by computing $\hat{\rho}^b = \left\{\hat{\rho}_j^b = \beta_j^b \oplus s_j^{b\oplus\alpha_j}\right\}_{j\in\Delta}$ for $b \in \{0,1\}$ (The rationale behind this extraction strategy is that, for every input b, the sender of the protocol Π_{OT} is supposed to send "encryption" $\left\{\beta_j^b\right\}$ of the shares $\left\{\rho_j^b\right\}$ of that input in Stage 5, hidden using the appropriate inputs $\left\{s_j^{b\oplus\alpha_j}\right\}$ of the OT executions). Given the shares $\hat{\rho}^0$ and $\hat{\rho}^1$, Sim reconstructs inputs $\hat{v}^0$ and $\hat{v}^1$ as follows: For every b, it checks whether $\hat{\rho}^b$ is $16n$-close to a valid codeword $\hat{w}^b$, and whether $\hat{w}^b$ passes the consistency check in the last stage, that is, if $\hat{w}^b$ agrees with $\beta_j^b \oplus \hat{s}_j^{b\oplus\alpha_j}$ for all $j \in \Gamma$; If so, then it sets $\hat{v}^b$ to the value encoded in $\hat{w}^b$; otherwise it sets $\hat{v}^b = \perp$. Finally Sim sends the two values $\hat{v}^0$ and $\hat{v}^1$ externally to the OT functionality.

Strategy 3: If both the Sender (P_i) and the Receiver (P_j) are honest, the simulator needs to simulate the transcript of an honest execution of Π_{OT} for A without knowing the inputs of the honest players. To do so, it simply generates the transcript of an honest execution of Π_{OT} using inputs all 0.

Below we analyze each of the simulation strategies above, and show that the environment Z's interactions with S in the ideal-world is indistinguishable from that with A in the real-world in each of the cases.

Analysis of the first case: Consider the following five hybrids:

Hybrid H_1: Hybrid H_1 proceeds identically to the ideal execution, except that: In Stage 2, for every location j that the sender would not need to reveal the randomness, that is, $j \notin \Gamma_S \cup \Gamma$ (recall that the simulator obtains Γ_S and Γ by forwarding A's commitments to the helper functionality $\mathcal{H}$ at the end of Stage 1), the simulator simulates the j^{th} commitment of $\langle C, R\rangle$ to A by committing to 0 instead of a_j^S. Since these commitments all have identities belonging to an honest player, and the helper functionality $\mathcal{H}$ only breaks

commitments with identities of corrupted parties, it follows from the CCA-security of $\langle C, R\rangle$ that the ideal-execution is indistinguishable from H_1.

Hybrid H_2: Hybrid H_2 proceeds identically to H_1 except that in Stage 5, instead of always using the sender's inputs s_j^0 and s_j^1 for $j \in \Delta$ to hide the shares $\rho^0 = \{\rho_i^0\}_{i\in\Delta}$ and $\rho^1 = \{\rho_i^1\}_{i\in\Delta}$, the simulator replace those inputs s_j^b's that are computationally hidden from A with random strings to hide the shares. More precisely, recall that in every location $j \in \Delta - \Phi - \Gamma$, A acts honestly in the OT execution (using input c_j) and the sender's inputs are not revealed in Stage 6; thus the input $s_j^{1-c_j}$ is computationally hidden. Then, instead of using $s_j^{1-c_j}$ as a one-time pad to hide one of the j^{th} shares ρ_j^0 or ρ_j^1 in Stage 5, Sim uses a truly random string.

We claim that H_2 is indistinguishable from H_1. Towards showing this, we first show that it follows from the security of the OT protocol $\langle S, R\rangle$ against semi-honest receiver that the views of a *malicious* receiver R^* in the following two experiments are indistinguishable.

> In both experiments, the receiver R^* on input a choice u, random input τ and auxiliary input z, first engages in an execution with the honest sender S with inputs s_0 and s_1 chosen at random. After the execution with S completes, R^* receives the two inputs s_0 and s_1 in the first experiment. On the other hand, what R^* receives in the second experiment depends on whether it has acted honestly according to inputs u and τ: If it is dishonest, then it still receives s_0 and s_1; otherwise, if it is honest, it receives s_u and a random bit s'.

It follows from the security against semi-honest receivers that when R^* acts honestly according to a choice u, the sender's input s_{1-u} that is not chosen is computationally hidden, and thus R^* cannot tell apart later whether it receives s_{1-u} or a random bit. Therefore its views in the above two experiments are indistinguishable. It further follows from a simple hybrid argument that, no malicious receiver can tell apart the two experiment even if they are executed in parallel.

Then, since the only difference between hybrid H_1 and H_2 lies in whether the sender's message in Stage 5 is generated using honest inputs $s_j^{1-c_j}$ or random strings, for those locations $j \in \Delta - \Phi - \Gamma$ where the adversary A acts honestly in the OT execution according to the inputs and random tapes decided in Stage 2. It then follows from the indistinguishability of the above two experiments (executed in parallel) that the sender's messages in Stage 5 in H_1 and H_2 are indistinguishable. Then, by the 1-robustness of the CCA-secure commitment $\langle C, R\rangle$, we have that the view of A in the two hybrids are indistinguishable. Thus H_1 and H_2 are indistinguishable.

Hybrid H_3: This hybrid proceeds identically to H_2 except the following: The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ does not expect to receive a choice from the simulator and instead directly discloses both of the sender's inputs v_0 and v_1 to the simulator; then after the simulator extracts the adversary's choice u, instead of using inputs v_u and a random bit to simulate the Stage 5 message as in H_2, the simulator uses v_0 and v_1.

To show that H_3 is indistinguishable from H_2, we first prove that due to the cut-and-choose procedure in Stage 4, the probability that A cheats in a large number of—more than n—OT executions in Stage 3 without being caught is negligible.

Claim 4. *In hybrid H_4, the probability that A cheats in more than n OT executions in Stage 3, and successfully passes the receiver's honesty check in Stage 4 is negligible.*

We remark that this claim holds, even if A receives a commitment to the set of locations Γ_R to be opened in the cut-and-choose procedure in Stage 1. This is because the CCA-

security of the commitment guarantees that Γ_R remains hidden even if all the values that A commits to in Stage 2 are extracted via a committed-value oracle (since these commitments use identities of corrupted parties, different from the identity of the commitment to Γ_R, which is the identity of the honest sender.) Then since we can identify the locations where A cheats using these committed values efficiently, these locations must be computationally independent of Γ_R. Thus if A cheats in a large number of OT executions, one of them will be selected by Γ_R to check with overwhelming probability, causing A to fail in Stage 4. A formal proof of this claim is presented at the end of this section.

It follows directly from Claim 4 that except with negligible probability, if A completes Stage 4 successfully, then the total number of shares that it gets is at most $20n$; then, by the pigeon hold principle, it gets more than $10n$ shares for at most one input. This implies that the probability that the simulator outputs fail is negligible. Assume that S_4 does not output fail. Then the only difference between the simulation in hybrid H_2 and H_3 lies in how the Stage 5 sender's message is simulated. In H_2, for the input that A gets more than $10n$ shares, the simulator obtains the true value through the OT functionality and thus simulate the corresponding part of the sender's message perfectly. On the other hand, for the inputs that A gets no more than $10n$ shares, it simulates shares of these inputs using shares of random bits. However, since A gets at most $10n$ shares of these random bits and the rest of shares are all covered by truly random strings in H_2, switching the random bits to true inputs does not change the distribution of the simulated message (of Stage 5). Thus we conclude that the views of A in H_3 and H_4 are statistically close, and so are the executions of H_3 and H_4.

Hybrid H_4: Hybrid H_4 proceeds identically to H_3 except that in Stage 5, the simulator switches back to using the sender's inputs s_j^0 and s_j^1 in the OT executions to hide the shares $\rho^0 = \left\{\rho_j^0\right\}_{j\in\Delta}$ and $\rho^1 = \left\{\rho_j^1\right\}_{j\in\Delta}$ (instead of using random strings to hide part of the shares). In other works, H_4 reverses the changes done in H_2. Then it follows from the same argument as in H_2 that, by the 1-robustness of the commitment $\langle C, R\rangle$, the hybrid H_4 proceeds indistinguishably from H_3.

Hybrid H_5: Hybrid H_5 proceeds identically to H_4 except that, in Stage 2, for every location j that the sender do not need to reveal the randomness, that is, $j \notin \Gamma_S \cup \Gamma$, the simulator switches the j^{th} commitment of $\langle C, R\rangle$ to A from committing to 0 back to committing to a_j^S. That is, H_4 reverses the changes done in H_1. Then it follows from the same argument as in H_1 that by the CCA-security of $\langle C, R\rangle$, the hybrid H_5 proceeds indistinguishably from H_4.

Finally note that in H_5, the simulator receives from $\mathcal{F}_{\mathsf{OT}}$ both inputs v_0 and v_1, and internally emulates the real-execution with A perfectly. Therefore, by a hybrid argument, we have that the real execution is indistinguishable from an execution of H_1, which in turn is indistinguishable from from the ideal execution. Thus we concludes that the simulator Sim is constructed correctly.

Analysis of the second case: Consider the following sequence of hybrids.

Hybrid $\tilde{H}_1$: This hybrid proceeds identically to the ideal execution, except the following: In Stage 2, for every location j that the receiver do not need to reveal the randomness, that is, $j \notin \Gamma_R$ (recall that the simulator obtains Γ_R by forwarding A's commitments to the helper functionality $\mathcal{H}$ at the end of Stage 1), the simulator simulates the j^{th} commitment of $\langle C, R\rangle$ from the receiver to A, by committing to 0 instead of a_j^R. Since

these commitments all have the identity of the honest receiver, and the helper functionality $\mathcal{H}$ only breaks commitments with identities of corrupted parties, it follows from the CCA-security of $\langle C, R\rangle$ that the ideal-execution is indistinguishable from $\tilde{H}_1$.

Hybrid $\tilde{H}_2$: This hybrid proceeds identically to $\tilde{H}_1$ except the following: the ideal functionality $\mathcal{F}_{\mathsf{OT}}$ discloses the external receiver's choice u to the simulator; furthermore, in Stage 5, instead of simulating all the α_j's using random bits, the simulator computes them honestly as $\alpha_j = u \oplus c_j$ (where c_j's are the inputs that the receiver uses in the Stage 3 OT executions).

We claim that $\tilde{H}_2$ is indistinguishable from $\tilde{H}_1$. Towards showing this, we first show that it follows from the security of the OT protocol $\langle S, R\rangle$ against malicious senders that the views of a malicious sender S^* in the following two experiments are indistinguishable.

> In both experiments, the sender S^* first participates in an interaction with an honest receiver R using a random inputs u. After the execution with R, in the first experiment, S^* receives the input u, whereas in the second experiment, it receives another independently sampled random bit u'.

It follows from the security against malicious senders of the OT protocol that the views of the malicious sender S^* in the above two experiments are indistinguishable. Furthermore, it follows from a simple hybrid argument that, no malicious receiver can tell apart the above two experiments even if they are executed in parallel.

Then, note that the only difference between hybrid $\tilde{H}_1$ and $\tilde{H}_2$ lies in whether in Stage 5, the α_j's (for $j \in \Delta$) from the receiver are simulated using random bits or generated honestly as $u \oplus c_j$; in other words, the differenece is whether the α_j's are computed as the sum of u and a random bit (yielding a random bit) or c_j (yielding $u \oplus c_j$). It then follows from the indistinguishability of the above two experiments (executed in parallel) that the receiver's messages in Stage 5 in $\tilde{H}_1$ and $\tilde{H}_2$ are indistinguishable. Thus, by the 1-robustness of the CCA-secure commitment $\langle C, R\rangle$, we have that the view of A and the output of the external OT receiver (which will be $\hat{v}_u$) in the two hybrids are indistinguishable. Hence $\tilde{H}_1$ and $\tilde{H}_2$ are indistinguishable.

Hybrid $\tilde{H}_3$: This hybrid proceeds identically to $\tilde{H}_2$ except that, in Stage 2, for every location j that the receiver do not need to reveal the randomness, that is, $j \notin \Gamma_R$, the simulator switches the j^{th} commitment of $\langle C, R\rangle$ from the receiver to A from a commitment to 0 back to a commitment to a_j^S. That is, $\tilde{H}_3$ reverses the changes done in $\tilde{H}_1$. Then it follows from the same argument as in $\tilde{H}_1$ that by the CCA-security of $\langle C, R\rangle$, the hybrid $\tilde{H}_3$ proceeds indistinguishably from $\tilde{H}_2$. We remark that in $\tilde{H}_3$ the simulator essentially emulates the execution of Π_{OT} for A perfectly as an honest receiver (using the external receiver's true input u that it gets from $\mathcal{F}_{\mathsf{OT}}$), except that it tries to extract the two sender's inputs from A at the end of the execution.

Hybrid $\tilde{H}_4$: This hybrid proceeds identically to $\tilde{H}_3$ except the following: The external receiver no longer interacts with $\mathcal{F}_{\mathsf{OT}}$, and instead, simply outputs a value that the simulator feeds it. On the other hand, the simulator internally emulates the execution of Π_{OT} for A by following the honest receiver's strategy (as in $\tilde{H}_3$), obtaining an output v; it then skips extracting the sender's inputs $\hat{v}_0$ and $\hat{v}_1$ and simply feeds the external receiver the value v as the its output.

We show that the output of the environement in $\tilde{H}_4$ is statistically close to that in $\tilde{H}_3$. Since the only difference between the two hybrids lies in how the outputs of the external receiver are derived, it suffices to show that except with negligible probability, the outputs of the external receiver in the two hybrids are the same. In $\tilde{H}_4$, the external receiver directly outputs the value v the simulator feeds it, which is just the output of

an honest receiver of Π_{OT} with input u (emulated by the simulator for A). In $\tilde{H}_3$, the external receiver obtains the u^{th} input from $\mathcal{F}_{\mathsf{OT}}$, which is $\hat{v}_u$ extracted by the simulator from A. Recall that both v and $\hat{v}_u$ are derived in two steps:

- In the first step, shares of the two values are recovered. The honest receiver obtains shares of v by computing $\tilde{\rho} = \{\tilde{\rho}_j = \beta_i^u \oplus \tilde{s}_i\}_{j\in\Delta}$, where $\tilde{s}_j$'s are the outputs it obtains in the Stage 3 OT executions with inputs c_j's. On the other hand, the simulator extracts shares of $\hat{v}_u$ as $\hat{\rho}^u = \left\{\hat{\rho}_j^u = \beta_j^u \oplus s_j^{u\oplus\alpha_j}\right\}_{j\in\Delta}$, where the s_j^b's are the inputs that A is supposed to use in the OT executions.
- In the second step, a codeword is recovered from the shares: The honest receiver recovers w that is $17n$-*close* to $\tilde{\rho}$, whereas the simulator recovers $\hat{w}^u$ that is $16n$-*close* to $\hat{\rho}^u$. Then both codewords are checked for consistency, that is whether they agrees with $\beta_j^u \oplus \hat{s}_j^{u\oplus\alpha_j}$ at locations $j \in \Gamma$, where the $\hat{s}_j^b$'s are A's actual inputs in the OT executions revealed in the last stage. If w (or $\hat{w}^u$ respectively) passes the consistency check, then v (or $\hat{v}_u$ resp.) is set to the value encoded in w (or $\hat{w}^u$ resp.); otherwise, it is set to $\perp$.

Towards showing that v and $\hat{v}_u$ are (almost) always the same, Consider the following two possible cases: $\tilde{\rho}$ is $17n$-close to a valid codeword w or not.

- If it is, (in $\tilde{H}_4$) the honest receiver will recover w from $\tilde{\rho}$. We show that the simulator will recover the same codeword w from $\hat{\rho}^u$ (i.e., $\hat{w}^u = w$). This relies on the following claim:

  Claim 5. *Let $\tilde{\rho}$ and $\hat{\rho}^u$ be defined as above. Then, except with negligible probability, the shares $\tilde{\rho}$ and $\hat{\rho}^u$ are $17n$-close to each other.*

  This claim essentially follows from the fact that due to the cut-and-choose procedure, the probability that A cheats in more than n OT executions in Stage 3, and yet, passes the sender's honesty check in Stage 4 is negligible. (This follows from the same proof as Claim 4). Therefore, there are at least $17n$ locations where A acted honestly (in the OT executions), meaning that the output $\tilde{s}_j$ of the honest receiver in these OT executions equals to $s_j^{c_j}$, which in turn equals to $s_j^{u\oplus\alpha_j}$ since $\alpha = u \oplus c_j$ in $\tilde{H}_3$ and $\tilde{H}_4$. This implies that $\tilde{\rho}$ and $\hat{\rho}^u$ agree with each other at at least $17n$ locations.

  Therefore, except with negligible probability, $\hat{\rho}^u$ is $16n$-close to w. Then the simulator will uniquely recover w as well (i.e., $\hat{w}^u = w$). After that, both the honest receiver and the simulator conduct the same consistency check against w, leading to the same outputs $v = \hat{v}^u$.
- Otherwise, if $\tilde{\rho}$ is n-far away from any valid codeword, the honest receiver sets v to $\perp$. We need to show that the simulator will also set $\hat{v}^u$ to $\perp$ with overwhelming probability. Suppose not, then the simulator must have recovered a codeword $\hat{w}^u$ from $\hat{\rho}^u$. By our hypothesis, $\hat{w}^u$ is n-far away from $\tilde{\rho}$; let Ψ be the set of locations j at which $\hat{w}^u$ and $\tilde{\rho}$ differ. Then we show that the probability that $\hat{w}^u$ passes the consistency check is negligible. Formally,

  Claim 6. *Let Ψ be defined as above. Then, the probability that $|\Psi| > n$ and A successfully passes the consistency check in Stage 6 is negligible.*

  This claim essentially follows from the fact that due to the cut-and-choose procedure, the probability that Ψ is large but none of the locations j in Ψ is checked in the last stage (i.e., $\Psi \cap \Gamma = \emptyset$) is negligible. (This follows from a similar proof as Claim 4). With overwhelming probability there is a location j in Ψ being checked, forcing A to reveal the inputs $\hat{s}_j^0$ and $\hat{s}_j^1$ it uses in that OT execution, which also reveals the

difference between $\tilde{\rho}_j$ and $\hat{w}_j^u$. That is,

$$\tilde{\rho}_j = \beta_j^u \oplus \tilde{s}_j = \beta_j^u \oplus \hat{s}_j^{c_j} = \beta_j^u \oplus \hat{s}_j^{u \oplus \alpha_j} \neq \hat{w}_j^u$$

Thus, except with negligible probability, the adversary fails to pass the consistency check, and $\hat{v}^u$ is set to $\perp$ as claimed.

By construction, in the last hybrid $\tilde{H}_4$, the view of A is emulated perfectly according to Π_{OT} and the output of the external OT receiver is the same as that of the honest receiver of Π_{OT}. Therefore, the output of the environment in hybrid $\tilde{H}_4$ is identically distributed to the real-execution. Then by a hybrid argument, we have that the real execution is indistinguishable to $\tilde{H}_1$, and thus the ideal execution. Thus the simulator *Sim* is constructed correctly.

Analysis of the third case: Consider the following sequence of hybrids:

Hybrid $\hat{H}_1$: This hybrid proceeds identically to the ideal execution except the following: The ideal $\mathcal{F}_{\mathsf{OT}}$ functionality discloses the inputs (v_0, v_1) and u of the external sender and receiver to A, and furthermore, the simulator switches the sender's and receiver's inputs that it uses for simulating the transcript of Π_{OT} (for A) from all 0 to $(0, v_1)$ and 0. It follows from the analysis of the first case above that when considering a semi-honest adversary acting as the receiver of Π_{OT} and using input 0, the adversary cannot tell apart if the honest sender is using inputs $(0, 0)$ or $(0, v_1)$. Thus the simulated transcripts in $\hat{H}_1$ and the ideal execution must be indistinguishable. Furthermore since no player is corrupted, the helper functionality $\mathcal{H}$ would not accept any query from the adversary, the indistinguishability of the simulated transcripts directly implies the indistinguishability of $\hat{H}_1$ and the ideal execution.

Hybrid $\hat{H}_2$: This hybrid proceeds identically to $\hat{H}_1$ except that the simulator switches the receiver's input that it uses for simulating the transcript of Π_{OT} from 0 to 1. It follows from the analysis of the second case above that when considering a semi-honest adversary acting as the sender of Π_{OT}, the adversary cannot tell apart if the honest receiver is using inputs 0 or 1. Thus the simulated transcripts in $\hat{H}_1$ and $\hat{H}_2$ must be indistinguishable. Therefore, it follow from a similar argument as in $\hat{H}_1$ that $\hat{H}_1$ and $\hat{H}_2$ are indistinguishable.

Hybrids $\hat{H}_3$ and $\hat{H}_4$: These two hybrids proceed identically to $\hat{H}_2$ except that the simulator switches the inputs that it uses for simulating the transcript of Π_{OT} from $((0, v_1), 1)$ in $\hat{H}_2$ to $((v_0, v_1), 1)$ in $\hat{H}_3$ and to $((v_0, v_1), u)$ in $\hat{H}_4$. It follows from the same argument as in $\hat{H}_1$ that hybrid $\hat{H}_3$ is indistinguishable to $\hat{H}_2$, and from the same argument as in $\hat{H}_2$ that hybrid $\hat{H}_4$ is indistinguishable to $\hat{H}_3$.

Finally, in the last hybrid, A receives the transcript of the execution of Π_{OT} using true inputs as in the real execution. Furthermore, since the outputs of the honest receiver in $\hat{H}_4$ is identical to that in the real execution (both equal to v_u), we have that the output of the environment in $\hat{H}_4$ is identically distributed to that in the real execution. Then by a hybrid argument we conclude that the real and ideal executions are indistinguishable.

Proof of Claim 4. Let Φ be the set of locations where A cheats in Stage 3. We note that Φ can be computed efficiently given the values that A commits to using $\langle C, R \rangle$ in Stage 2. Recall that the receiver's honesty check in Stage 4 requires the receiver to open all the commitments it sends in Stage 2 at locations in Γ_R. Therefore if $\Phi \cap \Gamma_R \neq \emptyset$, by the statistically binding property of $\langle C, R \rangle$, A would have been caught cheating. Thus if A successfully completes Stage 4, it must hold that $\Phi \cap \Gamma_R = \emptyset$. Now assume for contradiction that A cheats in more than n OT executions, but

successfully completes Stage 4 in H_4 (i.e., $|\Phi| > n$ and $\Phi \cap \Gamma_R = \emptyset$) with polynomial probability. Then we show that A can be used to violate the CCA-security of $\langle C, R \rangle$.

Consider a machine B that acts as a receiver of $\langle C, R \rangle$ externally with access to the committed-value oracle $\mathcal{O}$; internally, it emulates an execution of hybrid H_4 with A, by using $\mathcal{O}$ to implement the helper functionality $\mathcal{H}$, except the following: It forwards messages from the external committer C to A as the sender's commitment in Stage 1; then after Stage 3, it computes the set Φ using the values that A commits to in Stage 2 obtained through the helper functionality $\mathcal{H}$ as in H_4; finally, it simply outputs Φ and aborts. It follows from the construction that, when B externally receives a commitment to Γ_R—a randomly chosen set of size n—using the identity of the honest sender, it internally emulates the execution of H_4 perfectly up to Stage 3; then by our hypothesis, with polynomial probability, it outputs a set Φ of size greater than n, disjoint with Γ_R. However, on the other hand, if B externally receives a commitment to 0 (still using the identity of the honest sender), then the probability that B outputs a set of size greater than n that is disjoint with the randomly chosen Γ_R is exponentially small. Finally since all the commitments that B queries to the committed-value oracle $\mathcal{O}$ have identities belonging to a corrupted party, different from the identity of the honest sender, we have that B violates the CCA security of $\langle C, R \rangle$. □