Adaptive Hardness and Composable Security in the Plain Model from Standard Assumptions

Ran Canetti[*] Huijia Lin[†] Rafael Pass[‡]

Abstract

We construct the first general secure computation protocols that require no trusted infrastructure other than authenticated communication, and that satisfy a meaningful notion of security that is preserved under universal composition—*assuming only the existence of enhanced trapdoor permutations.* The notion of security fits within a generalization of the "angel-based" framework of Prabhakaran and Sahai (STOC'04) and implies super-polynomial time simulation security. Security notions of this kind are currently known to be realizable only under strong and specific hardness assumptions.

A key element in our construction is a commitment scheme that satisfies a new and strong notion of security. The notion, security against chosen-commitment-attacks (CCA security), means that security holds even if the attacker has access to an *extraction oracle* that gives the adversary decommitment information to commitments of the adversary's choice. This notion is stronger than concurrent non-malleability and is of independent interest. We construct CCA-secure commitments based on standard one-way functions, and with no trusted set-up. To the best of our knowledge, this provides the first construction of a natural cryptographic primitive having *adaptive hardness* from standard hardness assumptions, using no trusted set-up or public keys.

[*]Tel Aviv University, Email: `Canetti@tau.ac.il`
[†]Cornell University, E-Mail: `huijia@cs.cornell.edu`
[‡]Cornell University, E-Mail: `rafael@cs.cornell.edu`

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 ith 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], this paradigm offers a general and expressive approach that allows capturing a wide variety of requirements and situations in a natural and precise way. The idea is to say that a protocol π securely realizes f if running π emulates an idealized protocol I_f 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*) that interacts with parties running I_f. An advantage of the simulation paradigm is its expressiveness: It allows capturing a large variety of security properties in a natural and precise way, simply by formulating the appropriate f. This idea was informally articulated in [GMW91]. Many formulations of this paradigm were proposed, e.g. [GL90, Bea91, MR91, Can00, Gol04, PW01, Can01]. A proof that the [GMW91] construction satisfies the [Can00, Gol04] definition eventually appeared in [Gol04], demonstrating the realizability of this definition. We call this definition *basic security.*

Basic security indeed seems to be adequate in situations when the protocol is run in isolation. However, it does not provide sufficiently strong *composability* guarantees. Let us explain.

Composable security. A useful notion of security should provide guarantees even in settings where multiple protocols co-exist in the same system and potentially interact with each other, or in other words are *composed* to form a larger system. We distinguish three quite different (and incomparable) properties to consider in such settings:

Concurrent Multi-Instance Security: The security properties relating to the local data and outputs of the *analyzed protocol itself* should remain valid even when multiple instances of the protocol are executed concurrently and are susceptible to coordinated attacks against multiple instances.

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.

Environmental Friendliness: The security properties of *other, potentially unknown protocols* that co-exist in the same system should not be adversely affected by adding the analyzed protocol.

The simulation paradigm suggests a natural approach to formulating composable notions of security: Consider a protocol π that securely realizes a function f (i.e., π emulates the ideal protocol I_f), and let ρ be a protocol that uses subroutine calls to protocol π. Now, since the execution of π should look to an observer just the same as an execution of I_f, the behavior of ρ

should, intuitively, remain unchanged when each call to π is replaced by a call to I_f. Therefore, rather than analyzing the protocol ρ that uses (potentially multiple instances of) π, we might as well analyze the simpler system where each instance of π is replaced by an instance of I_f.

Making good of this intuitive approach turns out to be non trivial. Specifically, the definitions of [GL90, Bea91] were not shown to have any composability properties, whereas those of [MR91, Can00, Gol04] only guarantee *non-concurrent composability.* That is, the above three properties related to composable security are guaranteed only when the protocol ρ makes sure that the instances of π run in sequence, with nothing else happening in the rest of the system from the onset of the execution of each instance until all participants of this instance complete their respective local processing of π. This is a significant restriction.

UC security [Can01] gives a more stringent formulation of the simulation paradigm than basic security, providing a very strong composability property that implies all three composability requirements discussed above. But these strong properties come at a price: Many natural functionalities cannot be realized with UC security in the *plain model,* where the only set-up provided is 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 multi-instance extension of basic security [Lin04].

Security with super-polynomial simulators (SPS) [Pas03] 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." Protocols that realize practically any functionality with SPS security in the plain model were shown based on sub-exponential hardness assumptions [Pas03, BS05, LPV09].

Although SPS security is sometimes weaker than basic security, it often provides an adequate level of security. Furthermore, SPS security guarantees concurrent multi-instance security (with super-polynomial simulation). However, SPS security is environmentally friendly only in a very partial way: For other protocols in the systems, it only preserves those security properties that withstand super-polynomial time adversaries. Furthermore, SPS security is not closed under composition (protocol $\rho^{\pi/\phi}$ where each instance of π is replaced by an instance of ϕ is not guaranteed to emulate ρ with SPS security, even if π realizes ϕ with SPS security), 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 allow mitigating these shortcomings of SPS security. 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. (In spirit, these resources allow the simulator to "reverse engineer" the adversary.) It is not hard to see that, for any angel, angel-based security implies SPS security. Furthermore, akin to UC security, angel-based security is closed under composition. That is, if π emulates ϕ with respect to some angel then, for any ρ, $\rho^{\pi/\phi}$ emulates ρ with respect to the same angel. This means that angel-based UC security, with any angel, can be used as a basis for modular analysis of protocols. It remains to consider the "environmental friendliness" of this class of notions. Here too angel based security *may* provide an improvement over SPS security: For the other protocols in the system, any security property that holds with respect to adversaries that have access to the specific angel in use will be preserved when adding the analyzed protocol to the system. This means that different angels provide different levels of "environmental friendliness".

Furthermore, angel-based security is potentially realizable: A protocol realizing practically any ideal functionality with respect to some angel, in the plain model, are constructed in [PS04]. A different construction, based on a different angel, is given in [MMY06]. In [BS05] it is remarked that their protocol can be shown to be angel-based secure but the specific angel is not given. These protocols are very attractive indeed: They use no trusted set-up other than authenticated communication, and obtain a meaningful and composable security notion. However, they are all based on strong and non-standard hardness assumptions. This leaves open the following questions:

> Is it possible to realize general functionalities with SPS security in the plain model, under standard hardness assumptions?
>
> Can we show angel-based security with respect to some angel, under standard hardness assumptions? Further, can we show that this angel is environmentally friendly, at least for some class of protocols?

1.1 Our Results

We show an angel with which we can securely realize the ideal commitment functionality, $\mathcal{F}_{\mathsf{com}}$, in the plain model, assuming only one way functions. We then rely on known results [CLOS02, DBR90, IPS08] to construct a protocol for general functionalities. (Here we need to assume also existence of enhanced trapdoor permutations.)

In contrast with the angels considered so far in the literature, which maintain only global state on the system and are otherwise stateless, our angel is highly interactive and stateful. (In fact, dealing with such angels requires some modification of the original model of [PS04].) Still, security with respect to our angel implies SPS security.

Furthermore, we demonstrate that our angel provides a partial notion of environmental friendliness, which we refer to as *robustness*:[1] Any attack mounted on a *constant-round* protocol by an adversary that uses our angel can be carried out by a polytime adversary with no angels at all.[2] In fact, we rely on this robustness property to argue that the [CLOS02, DBR90] protocol (using $\mathcal{F}_{com}$) remains secure even with respect to our angel (so that we can rely on this protocol to realize general functionalities).

To formally present and prove our results, we also re-cast the model of [PS04] within the extended UC (EUC) framework of [CDPW07]. Roughly speaking, this framework is identical to standard UC security as in [Can00], except that all parties have access to an additional global entity. In [CDPW07], this entity is used to model global set-up such as a reference string or strong public-key infrastructure. Here, in contrast, we consider a global entity that, as in [PS04], interacts only with the corrupted parties (and the environement in the EUC framework). This means that the actual protocol uses no trusted infrastructure, and the global entity becomes a means for relaxing the security requirement. We call the global entity, $\mathcal{H}$ a *helper functionality* or an *angel*, and denote the corresponding notion of security $\mathcal{H}$-EUC security.

Main Theorem (Informally Stated): Assume the existence of enhanced trapdoor permutations. Then there exists a subexponential-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.

We emphasize that this is the first protocol to acheive any non-trivial notion of fully concurrent multi-instance security in the plain model and under standard assumptions, let alone to say closure under composition or environmental friendliness. In particular, it yields the first protocol achieving SPS-security in the plain model based only on standard assumptions.

[1] This notion of robustness is a strengthening of the notion of robustness considered in [LPV09].

[2] In fact, at the cost of increasing the round-complexity of our protocol, the angel can be modified to retain the security of any protocol with an *a-priori* bounded number of rounds.

Intuitively, our helper functionality (i.e., angel) will allow a party P to obtain the decommitment information for commitments *where P is the committer.* This will allow the simulator (i.e., the adversary we need to construct for interacting with the ideal protocol) to extract the decommitment information from commitments made by corrupted parties. Given such extraction help, simulation is relatively easy.

The main challenge is to make sure that the adversary will not be able to use this angel in order to break the security of other commitments, that were made by uncorrupted parties. For this we need commitment schemes that allow a committer to commit securely even when the receiving party has access to an extraction oracle. We call this strong property *security against adaptive chosen commitment attack (CCA security).*

1.2 The Main Tool: CCA-Secure Commitments

The protocols of [PS04, MMY06] for angel-based UC security rely on certain, quite specific *adaptive hardness* assumptions, namely assumptions that postulate security even in face of adversaries that have adaptive access to some help information. Indeed, such form of adaptive hardness seems to be inherent in the angel-based approach to security. However, such assumptions appear to be qualitatively stronger than non adaptive ones. A natural question is then whether such an adaptive hardness property can be based on a more standard assumption.

We answer this question positively: We formulate an angel that provides a useful adaptive hardness property, and show that this adaptive hardness property can be based on a standard assumption, specifically existence of one way functions.

The adaptive hardness property is the CCA security property of commitment schemes mentioned above. 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 provides decommitments to commitments using any tag $\mathsf{id}' \neq \mathsf{id}$. CCA-security can be viewed as a natural strengthening of *concurrent non-malleability* [DDN00, PR03, LPV08]—roughly speaking, a commitment scheme is concurrently non-malleable if it is CCA-secure with respect to restricted classes of adversaries that only ask a single parallel—i.e., non-adaptive—decommitment query after completing its interaction with the committer.

It is not hard to construct CCA-secure commitments using trusted set-up (by e.g., relying on known constructions of CCA-secure encryption schemes). It is also quite simple to construct a CCA-secure commitment scheme under an adaptive hardness assumption (such as the existence of adaptively-secure one-way permutations—namely one-way permutations that remain uninvertible even if the adversary has access to a inversion oracle) [PPV08].[3]

Our main technical contribution consists of showing how to construct a CCA-secure commitment based only on one-way functions, and without any trusted set-up.

Theorem (informally stated) Assume the existence of one-way functions. Then, for every $\epsilon > 0$, there exists an $O(n^\epsilon)$-round CCA-secure commitment scheme (where n is the security parameter).

As far as we know this yeilds the first non-trivial primitive whose "adaptive hardness" can be proven based on standard assumptions without set-up. Note that many standard cryptographic primitives (such as signatures [GMR89], pseudo-random function [GGM86] and CCA-secure encryption [RS91]) consider adaptive attacks where an adversary has access to an oracle breaking the

[3][PPV08] considered a variant of the notion of CCA-security for non-interactive and perfectly binding commitment scheme (called adaptively-secure commitments). We have extended this definition to general commitment schemes.

primitive. However, all these cryptographic rely on some trusted set-up (in the case of signatures and CCA-secure encryption, the public-key used needs to be well-formed, whereas in the case of pseudo-random function, the "seed" needs to be uniform and perfectly hidden from the adversary).

Our construction of CCA-secure commitments Consider the scenario of *concurrent non-malleable commitments* [DDN00, PR03, LPV08]. We consider a man-in-the-middle attacker (MIM) that participates in one interaction "on the left" and many interactions "on the right". In the left interaction it receives a commitment, whereas in the right interactions it provides (potentially) many commitments, acting as a committer. Intuitively, we want to ensure that the values committed to on the right are "independent" of the value committed to on the left. Let us revisit the approach of [LPV08] (which builds on [DDN00]) for constructing such commitments. The basic idea is to have a protocol where the scheduling of the messages depends on the tag of the commitment. The scheduling ensure that for every right interaction with a tag that is different from the left interaction, there exists a point—called a safe-point—from which we can "rewind" the right interaction (and extract out the value committed to), without violating the hiding property of the left interaction. It now follow from the hiding property of the left interaction that the values committed to on the right do not depend on the value committed to on the left.

At first sight, it would seem that this type of construction and analysis could be directly used to guarantee also CCA-security: we simply view the extrenal interaction the adversary participates in as a receiver, as the left interaction, and view all the oracle calls as right interactions. Each time a right interaction completes, we "rewind" the appropriate safe-point to extract the decommitment information (without violating the hiding of the left interaction), and feed the decommitment information to the adversary.

But there is a problem with this approach. Recall that in the setting of CCA-security, we need to provide the adversary with the decommitment information 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. (Note that this does not happen in the context of concurrent non-malleability since in that setting it is suffices to extract the committed values at the end of the whole execution; it is, thus, enough to rewind the right interactions one at a time, in sequence.) The problem is analogous to the simulation problem in the context of *concurrent zero-knowledge* [DNS04], where a nesting verifier might cause the naive simulation to take exponential time. On a high-level, we resolve this problem using an idea that is similar to that used in the context of concurrent zero-knowledge in the the work of Richardson and Kilian [RK99]. We present a scheduling of messages (again based on the tag) which ensures that every right interaction has n^ϵ (where n is the security parameter and $\epsilon > 0$) safe-points, instead of just one. Intuitively, having many safe-points ensures that there is always at least one safe-point where the number of nested executions is "small," which in turn ensures that the overall running-time of the extraction is polynomial.

Formalizing this argument turns out to be tricky. The main obstacle is that the techniques used to acheive non-malleability are incompatible with those used in the context of concurrent zero-knowledge:

- In order to use the proof technique of [LPV08, DDN00], the protocol (roughly speaking) needs to consists of a sequence of *three-message* "slots" with the property that if the last two messages are rewound the committed value can be efficiently recovered, but if all three messages are rewound, then the committed value remains completely hidden. This is used to argue that there exist points where we can extract the committed value from the rigth interaction, without violating the hiding of the left interaction.

- Known concurrent zero-knowledge protocols [RK99, KP01, PRS02, PV08] (and their analyses) rely on the fact that the intial message in the protocol determines some value; this value can then be recovered by rewinding any "chunk" of more than two messages that does not contain the initial message[4]. This property does not hold for protocols such as [DDN00, LPV08] since they rely on the principle that three-message chunks reveal nothing when rewound.

To get around this problem we develop a new concurrent extraction technique (based on the simulator strategies in [RK99, PV08]) which can be applied also to protocols such as [DDN00, LPV08]. Roughly speaking, instead of determining what slot to rewind based on the number of new executions that started "within" the slot (as in [RK99, PV08]), we treat slots from different executions uniformly and instead decide whether to rewind a slot, based on the number of slots that are contained within it. (This idea is similar to one used in [DGS09] in a different context, but where a similar problem arises). This allows us to extract the decommitment information to all commitments provided by the adversary to its oracle, without violating the hiding property of the left interaction, and while ensuring that the (expected) running-time of the extraction procedure is polynomial.

Robust CCA-security "Plain" CCA-security only guarantees that the security of the particular commitment scheme in question remains intact when the adversary has access to an extraction oracle. As mentioned above, in our final construction we additionally require that an attacker having access to the extraction oracle should not be able to violate the security of *any* constant-round protocol; we call CCA-secure commitments satisfying this property *robust CCA-secure.* Robust CCA-security of our construction follows in essentially the same way as plain CCA-security. Roughly speaking, when extracting the commitments on the "right", we simply have to make sure not to rewind any messages from the constant-round protocol on the "left"; this is possible since the commitment scheme has a super constant number of slots.

2 Definition of CCA-Secure Commitments

2.1 Notations and Preliminaries

Let N denote the set of all positive integers. For any integer $n \in N$, let $[n]$ denote the set $\{1, 2, \dots, n\}$, and let $\{0,1\}^n$ denote the set of n-bit long string; furthermore, let ε denote the empty string. We assume familiarity with the basic notions of *Interactive Turing Machines* [GMR89] (ITM for brevity) and interactive *protocols.* Given a pair of ITMs, A and B, we denote by $\langle A(x), B(y)\rangle(z)$ the random variable representing the (joint) output of A and B, on common input z and private input x and y respectively, and when the random input to each machine is uniformly and independently chosen.

2.2 Commitment Schemes

A commitment scheme $\langle C, R\rangle$ consists of a pair of $\mathcal{PPT}$ ITMs C and R that interacts in a commit stage and a reveal stage. In this work, we consider commitment schemes $\langle C, R\rangle$ that are statistically binding and computationally hiding. Furthermore, We restrict our attention to commitment schemes where the reveal phase is non-interactive—the committer decommits to value v by simply sending a decommitment pair (v, d). We let $\mathsf{open}_{\langle C,R\rangle}$ denote the function that verifies the validity

[4]This is used in the analysis to ensure that the extraction for a particular execution does not get stuck because of some execution that started earlier.

of (v, d); the receiver accepts (v, d) if $\mathsf{open}(c, v, d) = 1$, and rejects otherwise, where c is the commitment, i.e., the transcript of messages exchanged in the commit phase. Additionally, we consider the following two properties of commitment schemes:

- We say that a commitment c is *accepting* if the receiver accepts the transcript c at the end of the commit stage, and *valid* if there exists a decommitment pair (v, d) such that $\mathsf{open}_{\langle C,R\rangle}(c, v, d) = 1$. Then we say that a commitment scheme is *efficiently checkable*, if every commitment that is accepting w.r.t the honest receiver is valid.
- A *tag-based commitment schemes* [PR05, DDN00] is a scheme where, in addition to the security parameter, the committer and the receiver also receive a "tag"—a.k.a. the identity—id as common input.

2.3 CCA-Secure Commitments

Security under chosen-ciphertext-attacks (CCA security) [RS91] has been studied extensively in the context of encryption schemes, where the confidentiality of encrypted messages is guaranteed even in the presence of a decryption oracle. We here define an analogous notion for commitment schemes. Roughly speaking, a commitment scheme is CCA-secure (chosen-commitment-attack) secure if the commitment scheme retains its hiding property even if the receiver has access to a a "decommiment oracle". Let $\langle C, R\rangle$ be a tag-based commitment scheme. A decommitment 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 n, chosen adaptively by A. At the end of each session, if the session is *accepting and valid*, it reveals a decommitment of that session to A; otherwise, it sends $\perp$. (Note that when a session has multiple decommitments[5], the decommitment oracle only to returns one of them. Hence, there might exist many valid decommitment oracles.)

Loosely speaking, a tag-based commitment scheme $\langle C, R\rangle$ is said to be CCA-secure, if there *exists* a decommitment oracle $\mathcal{O}$ for $\langle C, R\rangle$, such that, the hiding property of the commitment holds even with respect to adversaries with access to $\mathcal{O}$. More precisely, denote by $A^{\mathcal{O}}$ the adversary A with access to the decommitment oracle $\mathcal{O}$. Let $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{O}, A, n, z)$, where $b \in \{0, 1\}$, denote the output of the following probablistic 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 $\mathsf{id} \in \{0, 1\}^n$, 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 replace with $\perp$ 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).

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

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

[5]Note that the statistically binding property only guarantees that, with overwhelming probability, the committed value is unique. However, there may still exist many different decommitments.

We say that $\langle C, R\rangle$ is CCA-secure *if there exists a decommitment oracle $\mathcal{O}'$, such that, $\langle C, R\rangle$ is CCA-secure w.r.t. $\mathcal{O}'$.*

As mentioned in the introduction, CCA-security easily implies concurrent non-malleability [DDN00, PR03, LPV08]. Note that in the definition of CCA-security, we could also have considered an adversary that can select a pair of *sequences* $\vec{v}_0, \vec{v}_1$ of challenge messages (instead of simply a pair (v_0, v_1)). It follows using a standard hybrid argument that CCA-security implies security also in this setting.

2.4 k-Robust CCA-Secure Commitments

We introduce a strengthening of the notion of CCA-security analougosly to the notion of *robust non-malleable commitments* of [LP09]. We here 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 decommitment oracle.

Definition 2. *Let $\langle C, R\rangle$ be a tag-based commitment scheme, and $\mathcal{O}$ a decommitment oracle for it. We say that $\langle C, R\rangle$ is* k-robust CCA-secure w.r.t. $\mathcal{O}$, *if $\langle C, R\rangle$ is CCA-secure w.r.t. $\mathcal{O}$, and for every $\mathcal{PPT}$ adversary A, there exists a $\mathcal{PPT}$ simulator S, such that, for every $\mathcal{PPT}$ k-round ITMs B, the following two ensembles are computationally indistinguishable.*

- $\left\{\langle B, A^{\mathcal{O}}(z)\rangle(1^n)\right\}_{n\in N, z\in\{0,1\}^*}$
- $\left\{\langle B, S(z)\rangle(1^n)\right\}_{n\in N, z\in\{0,1\}^*}$

Thus, 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.

We say that a tag-based commitment $\langle C, R\rangle$ is robust CCA-secure if there exists a a decommitment oracle $\mathcal{O}$, such that, $\langle C, R\rangle$ is k-robust CCA-secure w.r.t. $\mathcal{O}$, for every constant k.

On the identity length Recall that in the definition of CCA-security, the adversary can pick arbitrary identities id for both the left and the right interaction. We may also consider a restricted notion of (robust) CCA-security where the adversary is restricted to use identities of some bounded length. As we show in the full version of the paper, standard techniques [DDN00] can be used to show that any *robust* CCA-secure commitment that is secure for identities of length $\ell(n) = n^{\varepsilon}$ can be turned into a robust CCA-secure commitment (that is secure for identities of length n).

Proposition 1. *Let ε be any constant such that $0 < \varepsilon < 1$, and $\langle C, R\rangle$ a robust CCA-secure commitment scheme secure for identities of length $\ell(n) = n^{\varepsilon}$. Then assume the existence of one-way functions, there exists a robust CCA-secure commitment scheme $\langle \hat{C}, \hat{R}\rangle$ secure for identities of length n.*

Proof. The transformation from a robust CCA-secure commitment $\langle C, R\rangle$ for short (n^{ε}-bit) identities into one for long identities (n-bit) uses standard techniques [DDN00] as follows: to commits to a message $v \in \{0,1\}^n$, the committer $\hat{C}$ on common input a security parameter $n \in N$ and an identity id $\in \{0,1\}^n$, first generates a key pair (sk, vk) of a signature scheme, such that the verification-key vk is of length n^{ε}[6], and sends vk and a signature of the n-bit identity id (using sk)

[6]The existence of signature schemes is implied by the existence of one-way functions [Rom90]. To get a signature scheme with a "short" verification-key, simply "scale-down" the security parameter.

to the receiver in the first stage. Then, in the second stage, it simply commits to v using $\langle C, R\rangle$ and the verification key vk as the identity.

Let $\mathcal{O}$ be a decommitment oracle of $\langle C, R\rangle$, with respect to which $\langle C, R\rangle$ is robust CCA-secure. Then consider a decommitment oracle $\hat{\mathcal{O}}$ of $\langle \hat{C}, \hat{R}\rangle$ that extends $\mathcal{O}$ in the following straightforward way: to decommit a commitment of $\langle \hat{C}, \hat{R}\rangle$, $\hat{\mathcal{O}}$ returns the decommitment that $\mathcal{O}$ returns for the Stage 2 commitment of $\langle C, R\rangle$. Below we show that $\langle \hat{C}, \hat{R}\rangle$ is robust CCA-secure w.r.t. $\hat{\mathcal{O}}$. First, it follows directly from the robustness of $\langle C, R\rangle$ w.r.t. $\mathcal{O}$ (and the fact that, given access to $\mathcal{O}$, the decommitment oracle $\hat{\mathcal{O}}$ can be emulated perfectly) that $\langle \hat{C}, \hat{R}\rangle$ is robust w.r.t. $\hat{\mathcal{O}}$. Then, for CCA security, consider an arbitrary adversary A that participates in an experiment of $\mathsf{IND}_b(\langle \hat{C}, \hat{R}\rangle, A, n, z)$. We claim that, except from negligible probabilty, A never picks the same verification key vk, as that picked in the left interaction (by the left committer), in any accepting right interaction that has a different identity from the left interaction. Otherwise, (using A,) we could construct an adversary, who with access to $\hat{\mathcal{O}}$, is able to forges a signature of a randomly chosen key (corresponding to the key chosen by the left committer); then, by the robustness of $\langle \hat{C}, \hat{R}\rangle$ w.r.t. $\hat{\mathcal{O}}$, there exists a simulator B that is able to forge a signature even without access to $\hat{\mathcal{O}}$, which violates the unforgibility of the signature scheme. In other words, every successful right interaction that has a different identity from the left, also has a different veficiation key from the left. It thus follows from the CCA-security of $\langle C, R\rangle$ that $\langle \hat{C}, \hat{R}\rangle$ is CCA-secure w.r.t. $\hat{\mathcal{O}}$. $\square$

3 Construction of a CCA-Secure Commitment

In this section, we show the following theorem.

Theorem 1. *Assume the existence of one-way functions. Then, for every $\epsilon > 0$, there exists an $O(n^\epsilon)$-round robust CCA-secure commitment scheme (where n is the security parameter).*

Our CCA-secure commitment $\langle C, R\rangle$ is based on a variant of the concurrent non-malleable commitment protocol of [LPV08], which in turn is based on message scheduling technique of [DDN00]. However, here we use a slightly different message schedule in order to provide more "safe" rewinding slots. We proceed to formally specifying the protocol. For simplicity of exposition, the description below relies on the existence of one-way functions with efficiently recognizable range, but the protocol can be easily modified to work with any arbitrary one-way function (see Remark 1 for more details). Furthermore, we also rely on 3-round special-sound proofs in our protocol, but the analysis also works also with 4-round proofs. (See Remark 2 for more details .)

Let ℓ and η be polynomials in the security parameter n. To commit to a value v, the Committer C and the Receiver R, on common input 1^n and the identity $\mathsf{id} \in \{0,1\}^{\ell(n)}$, proceeds in the following three stages in the commit phase.

- *Stage 1:* the Receiver picks a random string $r \in \{0,1\}^n$, and sends its image $s = f(r)$, through a one-way function f with an efficiently recognizable range, to the Committer. The Committer checks that s is in the range of f and aborts otherwise. Additionally, the receiver also sends the first messages r_1, r_2 for two commitments, using a a two-round statistically binding string commitment com.

- *Stage 2:* The Committer provides a commitment c_1 to v using the commitment scheme com and r_1 as the first message; let (v, d) denote the decommitment information. Next, it provides a commtiment c_2 to (v, d) using com and r_2 as first message. We refer to (r_1, c_1) as the *first commitment* and (r_2, c_2) as the *second commitment.*

- *Stage 3:* The Committer proves that
  - (r_1, c_1) is a valid commitment to v and (r_2, c_2) is a valid commitment to a decommitment pair for (r_1, c_1),
  - *or* s is in the image set of f.

  This is proved using $4\ell(n)\eta(n)$ invokations of a special-sound $\mathcal{WI}$ proof where the verifier query has length $3n$.[7] The messages in the proofs are scheduled based on the identity id and relies on scheduling pairs of proofs according to schedules design_0 and design_1 depicted in Figure 1. More precisely, the proof stage consist of $\ell(n)$ phases. In phase i, the committer provides $\eta(n)$ sequential $\mathsf{design}_{\mathsf{id}_i}$ pairs of proofs, followed by $\eta(n)$ sequential $\mathsf{design}_{1-\mathsf{id}_i}$ pairs of proofs.

In the reveal phase, the Committer simply decommits to the first commitment (r_1, c_1). The Receiver accepts if the decommitment is valid and rejects otherwise.

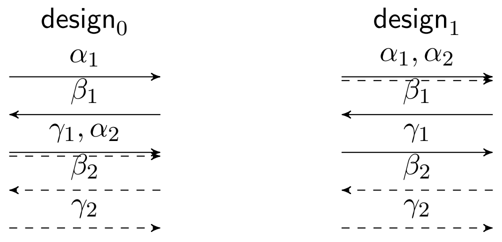


Figure 1: Description of the schedules used in Stage 3 of the protocol. $(\alpha_1, \beta_1, \gamma_1)$ and $(\alpha_2, \beta_2, \gamma_2)$ are respectively the transcripts of a pair of 3-round special-sound proofs.

On the round complexity of $\langle C, R\rangle$: The round complexity of $\langle C, R\rangle$ is $O(\ell(n)\eta(n))$. We will show that $\langle C, R\rangle$ is robust CCA-secure when $\eta(n) = n^{\epsilon}$ for any constant $\epsilon > 0$. When $\ell(n) = n^{\varepsilon'}$ (which is without loss of generality by Proposition 1) we thus obtain a $O(n^{\varepsilon+\varepsilon'})$-round protocol.

It follows using standard techniques that $\langle C, R\rangle$ is a commitment scheme with efficient verifiability.

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

Proof. It follows using essentially the same proof as in [LPV08] that the protocol is statistically binding and computationally hiding; we refer the reader to [LPV08] for more details.

It remains to show that the validity of an $\langle C, R\rangle$ commitment is efficiently checkable. By construction, an $\langle C, R\rangle$ commitment is valid if and only if the "first commitment" is valid. It then follows from the soundness of Stage 3 of the protocol that, whenever the receiver is accepting (at the end of the commit phase), the first commitment is valid except with negligible probability. Thus, the validity of $\langle C, R\rangle$ is also efficiently checkable. □

We turn to show that $\langle C, R\rangle$ is a robust CCA-secure commitment when $\eta(n) = n^{\epsilon}$ for any constant $\epsilon > 0$.

Theorem 2. *Let $\epsilon > 0$ be a constant, and let $\eta(n) = n^{\epsilon}$. Then $\langle C, R\rangle$ is a robust CCA-secure commitment.*

[7] As we shall see later on, the length restriction will facilitate the security proof.

To show that $\langle C, R\rangle$ is a robust CCA-secure commitment, we need to exhibit a decommitment oracle $\mathcal{O}$ for $\langle C, R\rangle$ such that $\langle C, R\rangle$ is robust CCA-secure w.r.t. $\mathcal{O}$. Consider the following decommitment oracle $\mathcal{O}$: $\mathcal{O}$ acts just as an honest receiver during the commit phase. At the end of every accepting interaction, if the "second commitment" (r_2, c_2) defines a unique value (v, d) such that (v, d) is a valid decommitment for the "first commitment" (r_1, c_2), $\mathcal{O}$ returns (v, d). Otherwise, $\mathcal{O}$ returns the lexicographically first decommitment, or $\perp$ if there does not exists a valid decommitment. The main technical challenge consists of proving the following proposition.

Proposition 3. *$\langle C, R\rangle$ is robust CCA-secure w.r.t $\mathcal{O}$.*

We provide a high-level overview of the proof below. The formal proof can be found in the full version of the paper.

Proof overview of Proposition 3 We first argue that $\langle C, R\rangle$ is CCA-secure w.r.t. $\mathcal{O}$. Consider the CCA-experiment IND_b, where the adversary A interacts with an honest committer C, and is given access to $\mathcal{O}$. We refer to its interaction with C as the *left interaction*, and its intearctions with $\mathcal{O}$ as the *right interactions*. 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 $\mathcal{O}$). The main hurdle in showing this is that the oracle $\mathcal{O}$ is not efficiently computable; if it was, indistinguishability would directly follow from the hiding property of the left interaction. However, since $\langle C, R\rangle$ consists of a sequence of special-sound proofs of the committed vaule, the oracle $\mathcal{O}$ can be efficiently implemented by "rewinding" the special-sound proofs in the right interaction. But, the problem is that once we start rewinding the right interactions, A might send new messages also in the left interaction. So, if done naively, this would require us to also rewind the left interaction, which could violate its hiding property.

The crux of the proof is showing how to appropriately rewind the right interactions (so as to extract out the commited values), without violating the hiding property of the left interaction. This implies that the view of A in IND_b can be efficiently emulated, without the oracle $\mathcal{O}$, and hence by the hiding property of the left interaction, A's view is indistinguishable. Towards doing this, we rewind the right interaction only at special points in the interaction; we call such points safe points.

Note that the hiding property of the left interaction remains intact if during the rewinding, one of the following two cases occurs.

- *Case 1:* A does not request any new messages in the left interaction.
- *Case 2:* The only new messages A requests in the left interaction are *complete $\mathcal{WISSP}$* proofs.

The fact that the left interaction is still hiding even if case 2 happens follows using exactly the same proof as proof of hiding of $\langle C, R\rangle$. And, obviously, the left interaction remains hiding if case 1 happens.

Roughly speaking, we now say that a prefix ρ of a transcript Δ (which consists of one left interaction and many right interaction) is a safe-point for the k^{th} right interaction if ρ "lies in between" the first two messages α_r and β_r of a $\mathcal{WISSP}$ proof $(\alpha_r, \beta_r, \gamma_r)$ for the kth interaction (i.e., it contains α_r but not β_r), and satisifes that from ρ to the point when γ_r is sent, one of the two cases above occurs in the left interaction. We call $(\alpha_r, \beta_r, \gamma_r)$ the *$\mathcal{WISSP}$ associated* with the safe point ρ in Δ. It follows using a combinatorical argument (similar in spirit, but more complicated than, [DDN00, LPV08]) that the message scheduling in Stage 3 of $\langle C, R\rangle$ guarantees the following key property:

Let Δ be any transcript of one left interaction, and many right interactions, of $\langle C, R\rangle$. Then, any right interaction k that 1) has completed, and 2) uses a different identity than the left interaction, has $\Omega(n^{\epsilon})$ *non-overlapping* $\mathcal{WISSP}$ that are associated with a safe point in Δ.

We will now use these safe-points to construct a simulator that can efficiently emulates the oracle $\mathcal{O}$. (As mentioned in the Introduction, the reason we need "many" safe-points is to ensure that we can extract out the committed values in all the right interactions while ensuring an expected polynomial running-time.) On a high-level, the simulation emulates $\mathcal{O}$ by following the honest receiver strategy of $\langle C, R\rangle$, until it enounters a "good" safe-point ρ. It then keeps rewinding the execution back to ρ until it obtains another accepting transcript of the right proof associated with ρ. Once two accepting proof transcripts are obtained, the special-soundness property allows the simulator to extract the decommitment information.

More precisely, the simulation is defined recursively in the following manner: On recursion level d, we say that a safe-point ρ of a transcript Δ is "good" (we call this a $d+1$-good safe-point) if the number of right-execution $\mathcal{WISSP}$ proofs—possibly from different interactions—starting in between ρ and the point where γ_ρ is sent in Δ, is smaller than $k_d = M/\eta'^{d+1}$, where M is a (polynomial) upperbound on the total number of messages in Δ, and $\eta' = n^{\varepsilon'}$ for some constant ε' such that $0 < \varepsilon' < \varepsilon$.

Then, on recursion level d, the simulator emulates every right interaction honestly, but as soon as it encounters a $d+1$-good safe-point in the current transcript, it begins "rewinding" the execution back to the safe-point, and invokes itself recursively at level $d+1$. In each rewinding, if it notices that ρ might no longer be a $d+1$-good safe-point, it cancels the rewinding and starts a new rewinding. It continues the process until it gets another transcript where ρ is a $d+1$-good safe-point again; from this second transcript we can extract out (and store) the decommitment information for the right interaction associated with the safe-point. Finally, whenever in the emulation a right interaction completes, the simulator provides A with the decommitment information extracted out (or outputs `fail` if the decommitment information has not been recovered). By cancelling "bad" rewindings (i.e., rewindings that are not $d+1$-good safe-points) we are guaranteeing two properties: 1) we *never* violate the hiding property of the left interaction, and 2) the running-time of the simulation does not blow up.

Let us now briefly argue that the simulator indeed emulates $\mathcal{O}$ both correctly and efficiently, without violating the hiding property of the left interaction.

Hiding property of the left interaction: Since the simulator only rewinds the right interactions from safe-points, and cancels every rewinding in which the point is no longer a safe-point, it follows that the left interaction remains hiding.

Correctness: We argue that for each right interaction that uses an identity that is different from the left interaction, we extract out the decommitment information before the interaction completes sucessfully. First, note that the recursion level is bounded by $c = \log_{\eta'} M$, which is a constant (since is M is polynomial in n and $\eta' = n^{\epsilon'}$). Since each successful right interaction, that uses a different identity than the left interaction, has n^{ϵ} safe-points (by the key property above), it follows that for each such interaction, there exists some recursive level d, such that the right interaction has at least $n^{\epsilon}/c > \eta'$ safe-points on level d. But, as the total number of right-proofs that start on level d is bounded by $k_d = M/\eta'^{d}$ (otherwise, the simulation at this recursive level is cancelled), there must exist one right-proof with an associated safe-point ρ, such that less than M/η'^{d+1} right-proofs start in between ρ and the last message of the proof. Therefore ρ is a $d+1$-good safe-point and will be rewound. Finally, since we continue rewinding until the decommitment information is found, it follows that for each successful right interaction that uses a different identity than the left interaction, the decommitment information is recovered by the simulator.

Efficiency: To prove that the simulation is efficient, consider a simplified scenario where A *never* sends a com commitment that can be decommitted to two different values—i.e., A never manages to violate the statistical binding property of com. We argue that the simulation is efficient in this case. (It suffices to consider this case, since the probability that A violates the statistical binding property of com is negligible, and thus we can always "cut-off" the simulation without loosing "too much".) To prove that the expected running-time of the simulation (in the simplified scenario) is polynomially bounded, first recall that the recursive depth is a constant c. Secondly, at each recursive level d, there are at most M possible points from which we can rewind and from each of these points, the expected number of rewindings is 1. The latter follows since the simulator only starts rewinds from a point ρ if it is a $\ell + 1$-good safe-point, and it continues rewinding until ρ becomes a $\ell + 1$-good safe-point again; furthermore, in each of the rewindings the simulated view of the adversary is identically distributed to its view in the first execution (this fact relies on us considering the case when all com commitments are well-defined). Thus, the probability that a point is a $\ell + 1$-good safe-point (conditioned on it occuring as a prefix in the execution) is the same in the first execution and in the rewindings. Therefore, the expected number of recursive calls starting from any point is 1. We now conclude that the expected total number of rewindings is bounded by $O(M)^c$.

The robustness w.r.t. $\mathcal{O}$ property of $\langle C, R\rangle$ follows using essentially the same proof as above: the key step here is, again, to show that the decommitment oracle $\mathcal{O}$ can be emulated efficiently, without "affecting" the security of the left interaction. Now, however, a rewinding is "safe" only if the adversary does not request *any* new message in the left (constant-round) interaction—that is, the the left interaction is never rewound during the extractions on the right. Roughly speaking, this is achieved by rewinding only those $\mathcal{WISSP}$ proofs that do not interleave with any message in the left interaction (and cancelling every rewinding in which the $\mathcal{WISSP}$ proof interleaves with a left-message).

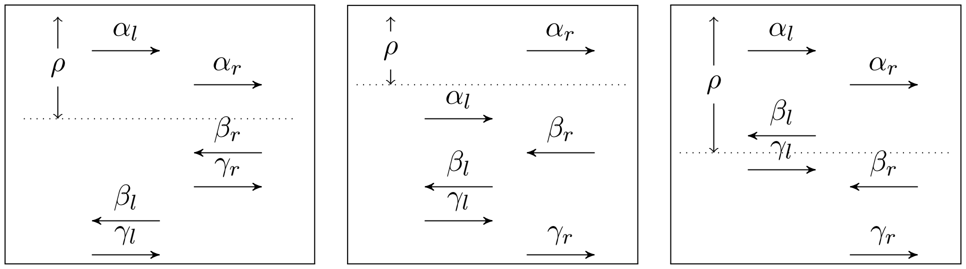
4 Proof of Proposition 3

We provide the formal definition of safe-points in the next section. The formal proof of Proposition 3 consists of two parts: in Section 4.2, we show that $\langle C, R\rangle$ is CCA-secure w.r.t $\mathcal{O}$; then in section 4.3, we show that it is also robust w.r.t. $\mathcal{O}$.

4.1 Safe-Points

Our notion of safe-points is almost the same as that in [LPV08] (which in turn is based on the notion of safe rewinding block of [DDN00]), with the only exception that our definition also considers the Stage 1 and 2 messages of the protocol, whereas the definition in in [LPV08] only concerns messages in the $\mathcal{WISSP}$ proofs.

Intuitively, a safe-point ρ of a right interaction k, is a point in Δ that lies in between the first two messages α_r and β_r of a $\mathcal{WISSP}$ proof $(\alpha_r, \beta_r, \gamma_r)$ 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 is sent, A expectes either no message or only complete $\mathcal{WISSP}$ proofs in the left interaction, as shown in Figure 2 (i) and (ii) respectively, (Additionally, in both cases, A may request the reply message of some $\mathcal{WISSP}$ proof, as shown in Figure 2 (iii). This is because, given the first two messages of a



(i) (ii) (iii)

Figure 2: Three characteristic safe-points.

$\mathcal{WISSP}$ proof, the reply message is deterministic, and hence can be emulated in the rewinding by replaying the reply in Δ.)

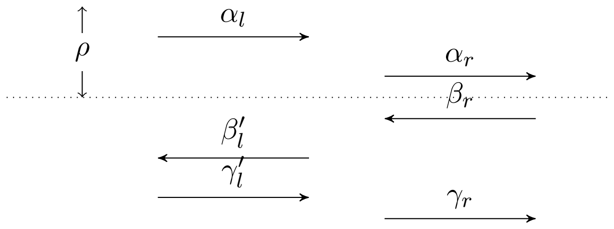


Figure 3: Prefix ρ that is not a safe point.

Definition 3. *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 proof $(\alpha_r, \beta_r, \gamma_r)$ in the right interaction k, such that:*

1. *α_r occurs in ρ, but not β_r (and γ_r).*
2. *for any proof $(\alpha_l, \beta_l, \gamma_l)$ in the left interaction, if α_l occurs in ρ, then β_l occurs after γ_r.*
3. *messages in Stage 1 and 2 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 proof associated with ρ. Note that the only case a right-interaction proof is not associated with any safe-point is if it is "aligned" with a left-execution proof, as shown in Figure 3. In contrast, in all other cases, a right-interaction proof has a safe-point, as shown in Figure 2. Below we show in Lemma 1 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 1 (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 $\mathcal{WISSP}$ *proofs that are associated with a* safe-point*.*

Proof. Consider a fixed $b \in \{0, 1\}$, and a transcript Δ of one left and many right interactions of $\langle C, R\rangle$ with A. We show below that in Δ, for every accepting right interaction that has a different identity from the left interaction, there are a number of $\Omega(\eta(n))$ $\mathcal{WISSP}$ proofs, $(\alpha_r, \beta_r, \gamma_r)$, such that there exists a point ρ—a prefix of Δ—satisfying the following two properties:

- ρ contains α_r, but not β_r and γ_r.
- For any proof $(\alpha_l, \beta_l, \gamma_l)$ in the left interaction, if α_l occurs in ρ, then β_l occurs after γ_r.

We call ρ a "weak" safe-point for $(\alpha_r, \beta_r, \gamma_r)$. It is easy to see that, if a right interaction has a number of $\mu = \Omega(\eta(n))$ proofs that are associated with a weak safe-point, then at least a number of $\mu/2$ of these proofs are completely sequentially arranged (as all the designs are sequentially arranged), and hence non-overlapping. Furthermore, since there are only a constant number of messages in Stage 1 and 2 of the protocol, at least $\mu/2 - c$ of these weak safe-points are actually full safe-points. Hence, we can conclude that there are $\Omega(\eta(n))$ non-overlapping proofs in this right interaction that are associated with a safe-point.

Now it only remains to show that there are $\Omega(\eta(n))$ proofs with a weak safe-point. Consider the following two adversarial schedulings in Δ:

- The adversary "aligns" the proofs in the left and right interactions *one by one*, where a left proof is aligned with a right proof if its challenge message lies in between the challenge and reply messages of the right proof in Δ.
- The adversary does not "align" the proofs in the left and right interactions.

We show that in the first case , there exist $\eta(n)$ weak safe-points; furthermore, we show that no matter how the adversary changes the message scheduling of the left interaction, the number of weak safe-points never decreases; hence, the claim also holds in the second case.

Assume that Case 1 holds. Since the identities of the left and right interactions, id^l and id^r, are different, they must exist one bit, i, on which they differ, i.e., $\mathsf{id}^l_i \neq \mathsf{id}^r_i$. Then each of the $\eta(n)$ design_1's in the i^{th} iteration (in Stage 3) of the right interaction k is aligned with a design_0 on the left. Below we show that whenever a design_1 is aligned with a design_0 on the left, there exists a weak safe-point: let $(\alpha^r_k, \beta^r_k, \gamma^r_k)$ for $k = 1, 2$ be the two proofs in the right design_1, and $(\alpha^l_k, \beta^l_k, \gamma^l_k)$ for $k = 1, 2$ the two proofs in the left design_0 that are aligned with them, as shown in figure 4. Then consider the prefix ρ of Δ that includes all the messages up to β^l_1. As β^l_1 lies in between β^r_1 and γ^r_1, so does ρ. Hence, ρ is associated with the right-proof $(\alpha^r_2, \beta^r_2, \gamma^r_2)$, and every the left proof either has its first two messages inside ρ, or after, which means ρ is a weak safe-point. Therefore each of the $\eta(n)$ design_1's in the i^{th} iteration of the right interaction has a weak safe-point.

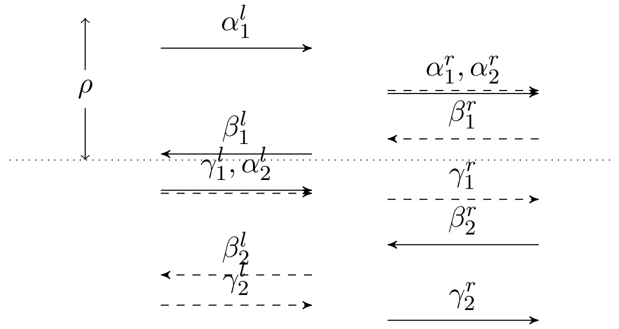


Figure 4: A design_0 matches up with design_1.

Next consider a game in which the adversary tries to decrease the number of weak safe-points to below $\eta(n)$, by changing the message-scheduling in the left interaction. By the argument above, we know that if the $\eta(n)$ design_1's in the i^{th} iteration of interaction j are aligned with design_0's, then there are at least $\eta(n)$ weak $\mathsf{safe\text{-}points}$, with one for each design_1. Then to succeed, the adversary must manage to eliminate the weak $\mathsf{safe\text{-}points}$ associated with $j > 0$ design_1's (in iteration i). Take any design_1 on the right.

- If one of the two proofs in the design_1 is not aligned with any proof on the left, then there exists a weak $\mathsf{safe\text{-}point}$ associated with that proof.
- If the two proofs in the design_1 are aligned with the two proofs in a design_0 on the left, then as argued above, there exists a weak $\mathsf{safe\text{-}point}$.
- If the two proofs in the design_1 are aligned with two left-proofs belonging to two different designs, then since the two left-proofs are sequentially arranged, as the two proofs in a design_0, it follows using the same argument as in the second case that there exists a weak $\mathsf{safe\text{-}point}$.

Therefore, the only way to ensure that a design_1 is not associated with any weak $\mathsf{safe\text{-}point}$ is to align it with a design_1 on the left. Then if the adversary wants to eliminate the weak $\mathsf{safe\text{-}points}$ associated with j design_1's in the i^{th} iteration on the right, namely designs $k_1^r < k_2^r < \ldots < k_j^r$, it must align them with j design_1's on the left, namely designs $k_1^l < k_2^l < \ldots < k_j^l$ (where k_h^l and k_h^r are the indexes of the designs). Since the $\eta(n)$ design_1's in the i'th iterations corresponds to design_0's on the left, it holds that for every h, either $k_h^l < k_1^r$ or $k_h^l > k_j^r$. Then there are only three possibilities, in which although the adversary may eliminate j weak $\mathsf{safe\text{-}points}$ associated with the design_1's in the i^{th} iteration on the right, it creates at least $2j$ new weak $\mathsf{safe\text{-}points}$ at other parts of the right interaction, namely,

The adversary shifts j left-designs down, i.e., $k_j^l < k_1^r$. Then the first $k_1^r - 1$ right-designs can only be aligned with the first $k_1^l - 1$ left-designs. Since $k_1^l - 1 \leq k_j^l - j \leq (k_1^r - 1) - j$, there are at least $2j$ proofs on the right (belonging to the first $k_l^r - 1$ right-designs) that are not aligned with any proofs on the left. Then each of them is associated with a weak $\mathsf{safe\text{-}point}$.

The adversary shifts j left-designs up, i.e., $k_1^l > k_j^r$. It follows from the same argument as above that there are at least $2j$ weak $\mathsf{safe\text{-}points}$ in the last $2\ell(n)\eta(n) - k_j^r$ designs on the right.

The adversary shifts some left-designs down and some up, i.e., $k_1^l < k_1^r$ and $k_j^l > k_j^r$. Then every right-design k, such that $k < k_1^r$ or $k > k_j^r$, can only be aligned with a left-design k', such that $k' < k_1^l$ or $k' > k_j^l$. Since there are at least $2\ell(n)\eta(n) - \eta(n)$ right-designs with indexes smaller than k_1^r or greater than k_j^r, but at most $2\ell(n)\eta(n) - \eta(n) - j$ left-designs with indexes smaller than k_1^l or greater than k_j^l, there are (again) at least $2j$ proofs on the right that are not aligned with any proofs on the left, which gives $2j$ weak $\mathsf{safe\text{-}points}$.

Therefore no matter how the adversary changes the message-scheduling in the left interaction, there always exist at least $\eta(n)$ weak $\mathsf{safe\text{-}points}$ in the right interaction. Hence we conclude the Lemma.

□

4.2 Proof of CCA Security

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

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

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

Towards this, we consider new commitment scheme $\langle \hat{C}, \hat{R}\rangle$ (similar to the "adaptor" schemes of [DDN00, LPV08]), which is a variant of $\langle C, R\rangle$ where the receiver can ask for an arbitrary number of special-sound $\mathcal{WI}$ designs in Stage 3. Furthermore, $\langle \hat{C}, \hat{R}\rangle$ does not have a fixed scheduling in Stage 3; the receiver instead gets to choose which design to execute in each iteration (by sending bit i to select design_i). 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 standard techniques 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,

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

We reach a contradiction by exhibiting a (stand-alone) adversary B^* that distinguishes commitments using $\langle \hat{C}, \hat{R}\rangle$. Let STA_b be defined identically to IND_b (in the definition of CCA-security), except that the adversary does not get access to a decommitment oracle. We show that the following two claims hold w.r.t B^*.

Claim 1. *There exists a polynomial T, such that, for every $n \in N$ and $z \in \{0,1\}^*$, the probability that the machine B^*, on input 1^n and z, in interaction with $\hat{C}$, runs for more than $T(n)$ steps is smaller than $1/3p(n)$.*

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

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

By Claim 2, it thus follows that for infinitely many $n \in N$, there exists $z \in \{0,1\}^*$, such that,

$$\left|\Pr\left[D(\mathsf{STA}_0(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)) = 1\right] - \Pr\left[D(\mathsf{STA}_1(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)) = 1\right]\right| \geq \frac{2}{3p(n)}$$

Finally, by Claim 1, the execution of B^* can be truncated after $T(n)$ steps, while only affecting the distinguishing probability by at most $\frac{1}{3p(n)}$, which means there exists a $\mathcal{PPT}$ machine that distinguishes commitments with probability $\frac{1}{p(n)}$; 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 decommitment oracle $\mathcal{O}^*$ for A by extracting the decommitments of the "right interactions" from the $\mathcal{WISSP}$ proofs in $\langle C, R\rangle$. It does this by rewinding A only at safe-points associated with the $\mathcal{WISSP}$ on the right. 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 decommitments 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 becomes exponential (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-proofs that start between ρ and the last message γ_ρ of the right-proof 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-proofs start between ρ and γ_ρ, where M is an upperbound 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 B^* 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-proof, a "repository" $\mathcal{R}$ of transcripts of right-proofs that have been previously collected, proceeds as follows:

Procedure $\mathsf{EXT}(d, \mathcal{V}, s, \mathcal{R})$**:** 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 sth right-proof. If at any point in the emulation, $d > 0$ and ρ is not a $d+1$-good safe-point, for s, EXT aborts returning $\perp$. 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., $d = 0$), when A is participating in a *complete* $\langle C, R\rangle$ commitment on the left, this can be easily done by simply requesting the appropriate designs from $\hat{C}$. At lower levels (i.e., $d > 0$), recall that 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 $\mathcal{WISSP}$ proofs; the former case can be trivially emulated (by simply doing nothing), in the latter case, EXT emulate the left interaction by 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 proof $(\alpha_r, \beta_r, \gamma_r)$ in a right interaction j, EXT attempts to extract a decommitment for this interaction, if the proof $(\alpha_r, \beta_r, \gamma_r)$ is associated with a $d+1$-good safe-point ρ' (for the transcript generated so far). 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 after the transcript ρ', and s' is the index of the right-proof $(\alpha_r, \beta_r, \gamma_r)$. It continues invoking itself recursively until one of the recursive invokations returns a view containing another accepting transcript $(\alpha_r, \beta'_r, \gamma'_r)$ of the s'-th proof. When this happens, and $\beta_r \neq \beta'_r$, EXTcomputes a witness w (by the special-soundness property of the $\mathcal{WISSP}$), and records w in the repository $\mathcal{R}$, if it contains the committed value (v, d) of the "second commitment", and (v, d) is a valid decommitment of the "first commitment" (and hence, by definition, also a valid decommitment). Later, whenever A expects a decommitment for a right interaction j that has an identity that is different from the left interaction, it simplys check the repository $\mathcal{R}$ for a matching decommitment; it aborts and outputs fail if no valid decommitment is available—we say that EXT "gets stuck" on interaction j in this case. (If A expects the decommitent of a right interaction that has the same identity as the left, it simply sends $\perp$ to A.)

We now return to specifying B^*. B^*, in interaction with $\hat{C}$, simply invokes EXT on inputs $(0, \mathcal{V}, \mathsf{null}, \emptyset, M, \eta')$, 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 interaction, 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 Claim 1—Running-time Analysis of B^*. Let $\overline{\mathsf{Bind}}$ denote the event that in an execution between B^* and $\hat{C}$, the adversary A provides a com commitment that has two valid decommitments. It follows from the statistically binding property of com, and the fact that B^* never runs for more than 2^n steps, that the event $\overline{\mathsf{Bind}}$ happens with only negligible probability (where the probability is taken over the randomness used by B^* and $\hat{C}$). Below we show the following claim.

Claim 3. *There exists a polynomial function t such that for every $b \in \{0,1\}$, $n \in N$ and $z \in \{0,1\}^*$, conditioned on the event that $\overline{\mathsf{Bind}}$ does not occur in the experiment $\mathsf{STA}_b(\langle \hat{C}, \hat{R} \rangle, B^*, n, z)$, B^* takes $t(n)$ steps in expectation.*

Claim 1 directly follow from Claim 3: Set $T(n) = 4p(n)t(n)$. By the Markov inequality we have that, conditioned on $\overline{\mathsf{Bind}}$ not occurring, the probability that B^* runs for more than $T(n)$ steps is at most $1/4p(n)$. Since $\overline{\mathsf{Bind}}$ occurs with only negligible probability, overall the probability that B^* runs for over $T(n)$ steps is at most $1/3p(n)$.

Proof of Claim 3. Towards bounding the running time of B^*, we construct another machine $\bar{B}$, which proceeds exactly the same as B^*, except that, it never aborts. More precisely, let $\mathcal{O}_{\mathsf{com}}$ be a oracle with *unbounded* running-time that on input a transcript of a com commitment returns the unique committed value. (It returns $\perp$ if the commitment is invalid or there does not exists a unique committed value.) Then consider a machine $\bar{B}$ that has access to $\mathcal{O}_{\mathsf{com}}$, and internally runs a variant $\overline{\mathsf{EXT}}$ of the procedure EXT: $\overline{\mathsf{EXT}}$ proceeds identically as EXT except that, once the execution "gets stuck" on a right interaction j, instead of aborting and outputting fail, it queries $\mathcal{O}_{\mathsf{com}}$ on the transcript of the second com commitment of interaction j, and uses the committed value returned from the oracle to continue the simulation; furthermore, $\bar{B}$ never cuts-off the execution of $\overline{\mathsf{EXT}}$ and may run for more than 2^n steps. It is obvious that $\bar{B}^{\mathcal{O}_{\mathsf{com}}}$ always runs longer than B^*. Therefore it is suffices to show that $\bar{B}$ runs in expected polynomial time, or equivalently, that $\overline{\mathsf{EXT}}$ runs in expected polynomial time, conditioned on $\overline{\mathsf{Bind}}$ not occurring. Below in Subclaim 1 we first show that the recursive depth of $\overline{\mathsf{EXT}}$ is a constant.

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

Proof. Let n^c be an upper bound on $M(n)$; and set D to $\lceil \log_{\eta'(n)} n^c \rceil$, which is a constant. Recall that the procedure $\overline{\mathsf{EXT}}(d, *, *, *, M, \eta')$ terminates and returns $\perp$ whenever more than $k_d = M/\eta'^d$ new right-proofs has started in its execution. When $d = D$, $k_D < 1$, which means the execution terminates whenever A starts a new right-proof. On the other hand, $\overline{\mathsf{EXT}}$ only makes a recursive call at the completion of a new right-proof. Therefore at recursion level D, $\overline{\mathsf{EXT}}$ never makes any recursive calls. $\square$

Next, assume that $\overline{\mathsf{Bind}}$ does not occur, that is, A never sends any com commimtment that has more than one decommitments. we show that the expected number of queries that $\overline{\mathsf{EXT}}$ makes to A at every recursion level $d \leq D$ is bounded by a polynomial. Then so is the expected running time of $\overline{\mathsf{EXT}}$.

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

Proof. We prove the subclaim by induction on d. When $d = D$, the claim follows, since $\overline{\mathsf{EXT}}$ does not perform any recursive calls and the number of queries made by $\overline{\mathsf{EXT}}$ can be at most the total number of messages, which is M.

Assume the claim is true for $d = d' + 1$. We show that it holds also for $d = d'$. Consider some fixed $\mathcal{V}, s$ and $\mathcal{R}$. The procedure $\overline{\mathsf{EXT}}(d', \mathcal{V}, s, \mathcal{R}, M(n), \eta'(n))$ simulates an execution with A in a straight-line on recursion level d', until it encounters the completion of a right-proof s that has a $d' + 1$-good $\mathsf{safe\text{-}point}$ ρ, then it tries to extract a witness of s, by repeatedly invoking $\overline{\mathsf{EXT}}$ on recursion level $d' + 1$ from (the partial transcript) ρ. Hence, the number of queries made by $\overline{\mathsf{EXT}}$ 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-proof s.Furthermore we compute the expected number of queries made by the recursive calls for a right-proof 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-proof s, conditioned on ρ occurring on level d'. Then

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

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', $\overline{\mathsf{EXT}}$ starts recursive calls from ρ for the right-proof s, which happens if and only if the proof s completes without being cancelled, *and* ρ is a $d' + 1$-good $\mathsf{safe\text{-}point}$ for it. When this happens, $\overline{\mathsf{EXT}}$ repeatedly calls itself on recursion level $d' + 1$, until an invocation succeeds without cancelling. (Recall that $\overline{\mathsf{EXT}}$ never aborts, hence whenever an invocation does not cancel, it returns an accepting transcript of s.) 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 $d'+1$. Let $q^s_{d'}(\rho)$ denote the probability that conditioned on ρ occurring on level d', ρ is a $d' + 1$-good $\mathsf{safe\text{-}point}$ for the right-proof s in a recursive call (from ρ for s) on level $d' + 1$. We claim that $q^s_{d'}(\rho) \geq p^s_{d'}(\rho)$. This follows from the fact that the view of A after ρ on level d' is simulated identically to that in a recursive call from ρ on level $d' + 1$: on both levels d' and $d' + 1$, $\overline{\mathsf{EXT}}$ emulates messages in the commitments of $\langle C, R\rangle$ for A perfectly; and furthermore, whenever A expects a decommitment of a right interaction, $\overline{\mathsf{EXT}}$ sends it the value committed to in the second com commitment, obtained either through recursive calls or through the oracle $\mathcal{O}_{\mathsf{com}}$; since, by our assumption, A never sends any com commitment that has multiple decommtments, A always receives the same value on both level d' and $d' + 1$.

Therefore, 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, $\overline{\mathsf{EXT}}$ 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 $\overline{\mathsf{EXT}}$ on level $d' + 1$ is at most $\theta(d' + 1)$. Therefore, if u recursive invocations are made from ρ for a right proof 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,

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

□

□

Proof of Claim 2—Correctness of the Output distribution of B^*. We proceed to show that the output distribution of B^* is correct. This follows from the following two claims:

Claim 4. *For every $b \in \{0,1\}$, $n \in N$ and $z \in \{0,1\}^*$, conditioned on the event that $\overline{\mathsf{Bind}}$ does not occur and that B^* does not output* fail*, the output view of A by B^* in the experiment $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)$ is simulated perfectly as in the experiment $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{O}^*, A, n, z)$.*

Claim 5. *For every $b \in \{0,1\}$, $n \in N$ and $z \in \{0,1\}^*$, conditioned on the event that $\overline{\mathsf{Bind}}$ does not occur, the probability that B^* outputs* fail *in the experiment $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)$ is negligible.*

Combining Claim 4 and 5 with the fact that $\overline{\mathsf{Bind}}$ occurs with only negligible probability, we have that, except with negligible probability, the output view of A by B^* is simulated perfectly. Hence, $\left\{\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)\right\}$ and $\left\{\mathsf{IND}_b(\langle C, R\rangle, \mathcal{O}^*, A, n, z)\right\}$ are indistinguishable.

Proof of Claim 4. First note that, in $\mathsf{STA}_b(\langle \hat{C}, \hat{R}\rangle, B^*, n, z)$, B^* outputs the simulated view $\mathcal{V}_A$ returned by the procedure EXT at the top recursion level ($d = 0$), and, as in $\mathsf{IND}_b(\langle C, R\rangle, \mathcal{O}^*, A, n, z)$, the view is replaced with $\perp$ if A copies the identity of the left interaction, in some right interaction. Hence it suffices to show that in the case where A never copies the identity of the left interaction, EXT simulates the messages in the left and right interactions for A perfectly. By construction of EXT, all the messages belonging to the commitments of $\langle C, R\rangle$ (both on the left and right) are simulated perfectly. Furthermore, conditioned on that B^* does not output fail, whenever A expects a decommitment of a right interaction, B^* extracts successfully a pair (v, d) that is the value committed to in the second com commitment, and is a decommitment of the first com commitment of this right interaction; then further conditioned on that $\overline{\mathsf{Bind}}$ does not occur, the decommitment oracle $\mathcal{O}$ would have returned exactly the same value. Hence we conclude the Claim. □

Proof of Claim 5. Consider a fixed $b \in \{0,1\}$. By Claim 3, conditioned on $\overline{\mathsf{Bind}}$ not occurring, B^* runs in $t(n)$ steps in expectation in the experiment $\mathsf{STA}_b(\langle \hat{C}, \hat{R} \rangle, B^*, n, z)$. By construction, B^* outputs fail, if the procedure EXT runs for more than 2^n steps, or it "gets stuck" on a right interaction j; and the latter happens if the interaction j succeeds and has a different identity from the left interaction, but one of the following three cases occurs.

Case 1: None of the $\mathcal{WISSP}$ proofs in the right interaction is rewound.

Case 2: Some proof is rewound but the recursive calls invoked for this proof generates the same proof transcript as in the interaction.

Case 3: A witness is extracted from one of the proofs in the interaction, but it is not the value committed to in the second com commitment, or is not a valid decommitment of the first com commitment in the interaction.

Below for each of the cases where B^* outputs fail, we show that it occurs with negligible probability.

Analysis of the case where B^* runs for more than 2^n steps: By Subclaim 3, conditioned on $\overline{\mathsf{Bind}}$ not occurring, B^* runs in expected polynomial time. Therefore, by the Markov inequality, the probability that EXT runs for more than 2^n steps 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 proofs must be rewound. By Lemma 1, there exist a number of $\Omega(\eta(n))$ non-overlapping proofs in the right interaction j that has a $\mathsf{safe\text{-}point}$. Recall that in B^* (more precisely, in EXT), a right interaction may be carried out at multiple different recursion levels (through recursive calls); and at level d, B^* rewinds every proof 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 proofs with a $\mathsf{safe\text{-}point}$ start in interaction j. Since the total number of right-proofs 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-proof that has a $\mathsf{safe\text{-}point}$ ρ, such that there are less than $M/\eta'(n)^{d+1}$ right-proofs starting in between ρ and the last message of the proof. Therefore ρ is a $d+1$-good $\mathsf{safe\text{-}point}$ for this right-proof, and will be rewound.

Analysis of Case 2: We bound the probability that any challenge message is picked twice in whole execution of B^* to be negligible. Since B^* runs for at most 2^n steps, it picks at most 2^n challenges during the whole execution. Furthermore, the length of each challenge is $3n$. By applying the union bound, we obtain that, conditioned on B^* running within 2^n steps, the probability that a challenge β on level d' is picked again is at most $\frac{2^n}{2^{3n}}$, and hence, using the union bound again, the probability that *any* challenge in the execution is picked twice is at most $2^n \frac{2^n}{2^{3n}}$. Hence, overall, the probability that this case occurs is negligible.

Analysis of Case 3: 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^*. By the special-sound property, it follows that the witness w extracted must be a value r such that $f(r) = s$, where s is the first message in the right interaction j. Then we show that we can invert the one-way function f. More precisely, given A, n and z, we construct A^* that inverts f. A^* on input $y = f(r')$, emulates an execution of $A(z)$ internally, exactly as B^* does, except that (1) it "cuts" the execution off after $g'(n) = 2t(n)g(n)$ steps, and (2) it picks a random right interaction started in the first $g'(n)$ steps, and feeds y as the Stage 1 message in that interaction; finally, A^* outputs all the witnesses extracted from this interaction. Since A^* proceeds identically as B^* in the first $g'(n)$ steps, the probability that it inverts f in some right interaction is exactly the same as the probability that B^* does in the first $g'(n)$ steps. Furthermore, since B in expectation takes $t(n)$ steps, by the Markov inequality, the probability that it runs for more than $g'(n)$ steps is at most $1/2g(n)$. Hence the probability that A^* inverts f in some right interaction is at least $1/g(n) - 1/2g(n)$. With probability at least $1/g'(n)$, A^* guesses correctly the right interaction in which this happens, and thus it inverts y with probability at least $1/2g(n)g'(n)$.

□

4.3 Proof of Robustness

In this section, we extend the proof in the last section to show that $\langle C, R\rangle$ is also *robust* CCA-secure w.r.t. $\mathcal{O}$. Towards this, we need to show (in addition to that $\langle C, R\rangle$ is CCA-secure w.r.t. CO^*) that for every constant k, 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.

Given an adversary A, and a constant k, the construction of the simultor 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 decommitments 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 $\mathcal{WISSP}$ proofs in Stage 3 of the protocol to extract the decommitments, 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 $\mathcal{WISSP}$ proofs that do not interleave with any messages in the left interaction, and cancelling every rewinding in which the $\mathcal{WISSP}$ proof 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-proof (α, β, γ) if it includes all the messages in Δ up to α, 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 fowarding 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 rounds, there exist $\Omega(n^{\varepsilon})$ R-safe-point in every successful right interaction. Then, it follows from the same proof as in Claim 3 and Lemma 2 that, except from negligible probability, S runs in expected polynomial time, and that the output of S in interaction with B is indistinguishable from the output of $A^{\mathcal{O}}$ in interaction with B.

Remark 1. *The protocol $\langle C, R\rangle$ described in Section 3 uses a one-way function with efficiently recognizable range in its first stage. It can be modified to work with any arbitrary one-way function f as follows: in Stage 1, the receiver sends the images of* two *secrets, i.e., $s_1 = f(r_1)$ and $s_2 = f(r_2)$, followed by a proof that either s_1 or s_2 is in the range of f, using a resettable $\mathcal{WI}$ proof system [CGGM00] (the committer verifies the proof and aborts if it is not convincing); and in Stage 3 the committer proves that either it has committed to v honestly, or that one of s_1 and s_2*

is in the range of f. It follows using almost the same proofs as above that the modified protocol is a robust CCA-secure commitment scheme, except that it relies on the one-wayness of f and, additionally, the resettable $\mathcal{WI}$ property of the proof in Stage 1 that, the value extracted from A is the desired decommitment, despite that A is rewound during the extraction. (See Case 3 in the proof of Claim 5.)

Remark 2. *We further modify the protocol to work with 4-round special-sound proofs instead of 3-round special-sound proofs: in Stage 1, the receiver sends, in addition to the images of two secrets, the first message r of a 4-round special-sound proof; then in Stage 3, the committer and the receiver simply use the last three messges of a 4-round special-sound proof with the first message fixed to r, as a 3-round special-sound proof. It follows from the same proof as described above that the modified protocol is robust CCA secure, as both the $\mathcal{WI}$ and the special-soundness properties hold even if the special-sound proofs share the same first message.*

5 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 paties.

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 identifer 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 ITM 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 (PPT). An ITM is 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].)

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 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 infrastruture [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 $r = r_Z, r_A, r_1, r_2, \ldots$ 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 $\textsc{exec}_{\pi,A,Z}(k,z,r)$ when r is uniformly chosen. Let $\textsc{exec}_{\pi,A,Z}$ denote the ensemble $\{\textsc{exec}_{\pi,A,Z}(k,z)\}_{k\in N,z\in\{0,1\}^*}$.

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 4. *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 $\textsc{exec}_{\phi,\mathcal{S},Z} \approx \textsc{exec}_{\pi,A,Z}$.*

Definition 5. *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 6. *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 $\textsc{exec}_{\phi,\mathcal{S},Z} \approx \textsc{exec}_{\pi,\mathcal{D},Z}$.*

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.

[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.

6 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 environment and the corrupted ITMs. Then:

Definition 7. *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 functionalities, this relaxed notion of security suffices even when $T(n) = exp(n)$ [Pas03, 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 6 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. □

7 Realizing Any Functionality Using CCA-Secure Commitments

We here show how to realize any functionality by relying on our construction of CCA-secure commitments. Let $\langle C, R\rangle$ be a commitment scheme that is robust CCA-secure w.r.t. a decommitment oracle $\mathcal{O}$. Furthermore, assume that $\mathcal{O}$ can be computed in subexponential time; note that our construction of CCA secure commitments can easily be instantiated with bit-commitments using a "scaled-down" the security parameter in order to ensure this property. Consider a helper functionality $\mathcal{H}$ that "breaks" commitments of $\langle C, R\rangle$ in the same way as $\mathcal{O}$ 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 decommitment oracle $\mathcal{O}$, and forwards P_i the decommitment pair returned from $\mathcal{O}$ at the end of each session. See figure 5 for a formal description of the functionality. Clearly this functionality can also be implemented in sub-exponential time.

Functionality $\mathcal{H}$

Initialization: Upon receiving an input (`Init`,P_i, sid, k) from party P_i in the protocol instance sid, if there is a previously recorded session (P_i, sid, i), 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, i).

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, i), ignore the message; otherwise, forward m to $\mathcal{O}$ in the i^{th} session that uses identity (P_i, sid), obtain a reply m', and return (`Mesg`,P_i, sid, i, m') to P_i.

Figure 5: The ideal functionality $\mathcal{H}$

We have now the following theorem:

Theorem 5. *Let ε be any positive constant. Assume the existence of enhanced trapdoor permutations. Then for every well-formed functionality*[9] *$\mathcal{F}$, there exists a $O(n^{\varepsilon})$-round protocol Π that $\mathcal{H}$-EUC-emulates $\mathcal{F}$.*

Towards proving the theorem, we first show how to implement the *ideal commitment functionality $\mathcal{F}_{\mathsf{com}}$* in the $\mathcal{H}$-EUC-model, and then show how to realize any functionality using $\mathcal{F}_{\mathsf{com}}$.

Realizing $\mathcal{F}_{\mathsf{com}}$: The ideal commitment functionality $\mathcal{F}_{\mathsf{com}}$, as presented in Figure 6, acts as a physical "lock-box" for the players in a secure computation: it enables a player (P_i) to commit to a receiver (P_j) a string m in a perfectly hiding way, by simply having P_i send m to the functionality in a commit phase; and later, in a reveal phase, it ensures that the commitment is opened in a perfectly binding way, by sending the unique previously recorded string m to P_j.

Lemma 2. *Let ε be any positive constant. There exists a $O(n^{\varepsilon})$-round protocol π_{com} that $\mathcal{H}$-EUC-emulates $\mathcal{F}_{\mathsf{com}}$.*

Proof. The committer P_i and the receiver P_j, on input (`Commit`, $(P_i, P_j, sid), v$) to P_i and (`Commit`, (P_i, P_j, sid)) to P_j, proceed as follows:

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

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

Commit Phase: Upon receiving an input $(\texttt{Commit}, sid, x)$ from C, verify that $sid = (C, R, sid')$ for some R, else ignore the input. Next, record x and generate a public delayed output $(\texttt{Receipt}, sid)$ to R. Once x is recorded, ignore any subsequent $\texttt{Commit}$ inputs.

Reveal Phase: Upon receiving an input $(\texttt{Open}, sid)$ from C, proceed as follows: If there is a recorded value x then generate a public delayed output $(\texttt{Open}, sid, x)$ to R. Otherwise, do nothing.

Figure 6: The ideal commitment functionality $\mathcal{F}_{\mathsf{com}}$

Stage 1: the receiver P_j picks a random secret $r \in \{0,1\}^n$, and commits to r using the CCA-secure commitment scheme $\langle C, R\rangle$, and using (P_j, sid) as the identity of the interaction.

Stage 2: the committer P_i commits to the value v using $\langle C, R\rangle$, and using (P_i, sid) as the identity of the interaction.

Stage 3: the committer P_i commits to 0^n using $\langle C, R\rangle$, and using (P_i, sid) as the identity of the interaction.

Finally, on input $(\texttt{Open}, (P_i, P_j, sid))$ to both P_i and P_j, the committer P_i decommits to the value v in a reveal phase, by sending v to P_j and then provides a strongly $\mathcal{WI}$ proof of the statement that either it has committed to v in Stage 2, or that it has committed to a valid decommitment pair (r, d) of the Stage 1 commitment, in Stage 3. The receiver P_j accepts if the proof is convincing, and rejects otherwise.

Next we proceed to show that π is indeed a secure realization of $\mathcal{F}_{\mathsf{com}}$. Below we describe the technique for simulating the protocol execution of π in the ideal-world, where parties have access to the ideal commitment functionality $\mathcal{F}_{\mathsf{com}}$, and give a proof that the simulation in the ideal-world setting is indistinguishable from a real-world execution of π. Recall that we only need to prove that π $\mathcal{H}$-EUC-emulates $\mathcal{F}_{\mathsf{com}}$; 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. The simulator S for A in the ideal world internally simulates a real-world execution with A: 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 the commit and reveal phases of the commitment $\langle C, R\rangle$ for A as follows:

Strategy 1: If the Committer (P_i) is honest and the Receiver (P_j) is corrupted, the simulator need to be able to complete the commit phase of the protocol on behalf of the committer without actually knowing the committed value v (which $\mathcal{F}_{\mathsf{com}}$ will not disclose until the reveal phase is initiated). Thus, the simulator needs to be able equivocate the commitment so that it can be opened to the proper value v in the subsequent reveal phase.

Towards this, in the commit phase, the simulator first forwards the $\langle C, R\rangle$ commitment from A (controlling P_j) in Stage 1 to the functionality $\mathcal{H}$. Since the receiver P_j is corrupted, $\mathcal{H}$ accepts commitments with identity P_j from S, and returns S a valid decommitment pair (r, d) if the commitment is accepting. (Note that it follows from the efficient verifiability property of $\langle C, R\rangle$ that if a commitment of $\langle C, R\rangle$ is accepting, then except from negligible probability, it is valid, and further by the statistically binding property, has a unique committed value, in which case, $\mathcal{H}$ would return a valid decommitment pair to this value.) Next the simulator completes the commit stage by committing to 0^n in Stage 2, and committing to the decommitment pair (r, d) in Stage 3. Then, later, in the reveal phase, S can open to any value v, by sending v to A and proving in the strongly $\mathcal{WI}$ proof that it has committed to a decommitment of the Stage 1 commitment in Stage 3.

Strategy 2: If the Committer (P_i) is corrupted and the Receiver (P_j) is honest, the simulator will need to learn the committed value v from the committer in order to correctly provide the corresponding commit phase input to $\mathcal{F}_{\mathsf{com}}$.

The simulator S emulates the messages from the receiver P_j for A (controlling P_i), by following the honest receiver strategy; additionally, it forwards the Stage 2 commitment of $\langle C, R\rangle$ from A to the decommitment functionality $\mathcal{H}$, and uses the committed value returned from the functionality as the commit phase input. More precisely, let (v, d) be the decommitment pair returned from $\mathcal{H}$; S then sends the message $(\texttt{Commit}, (P_i, P_j, sid), v)$ to $\mathcal{F}_{\mathsf{com}}$, if the Stage 2 commitment from A is accepting; otherwise, it does nothing.

Strategy 3: If both the Committer (P_i) and the Receiver (P_j) are honest, since the adversary A reads the communications between all parties, S needs to be able to simulate the commit phase between two honest parties for A, without knowing the value committed to by P_i, and later equivocate the simulated commitment to open to any value that P_i reveals. This task is almost the same as that in the first case, except that the receiver now is honest. Then S simulates the messages from the receiver P_j by following the honest receiver strategy; and simulates the messages from the committer P_i by applies Strategy 1 against the honest receiver strategy.

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 three hybrids:

Hybrid H_1 proceeds identically to the ideal-execution, except that, in H_1 the ideal commitment functionality $\mathcal{F}_{\mathsf{com}}$ discloses the commited value v to S, once it receives it from P_i; S then commits to v honestly, instead of 0^n, in Stage 2 of the commit phase simulated for A, using $\langle C, R\rangle$ and identity (P_i, sid). Then the only difference between the ideal-execution and H_1 lies in the value committed to in Stage 2 of the simulation of π. Since $\mathcal{H}$ "breaks" commitments of $\langle C, R\rangle$ in the same way as $\mathcal{O}$ does, it follows from the CCA-security w.r.t. $\mathcal{O}$ of $\langle C, R\rangle$ that, the execution in H_1 is indistinguishable from that in the ideal-world, provided that $\mathcal{H}$ does not "break" any commitments of $\langle C, R\rangle$ using identity (P_i, sid). (Recall that CCA-security holds only if the adversary does not query the decommitment oracle on any commitment using the same identity as the left interaction.) The last requirement holds, since $\mathcal{H}$ only accepts queries on commitments from the adversary or the environment, using their own identities, i.e., the identities of the corrupted parties; therefore, $\mathcal{H}$ never "breaks" any commitment with identity (P_i, sid) (which belongs to an honest party). Hence we conclude that the ideal-execution is indistinguishable from H_1.

Hybrid H_2 proceeds the same as H_1 does, except that S further emulates the reveal phase honestly for A, i.e., it proves in the strongly $\mathcal{WI}$ proof in the reveal phase that it has committed to v honestly in Stage 2 of the commit phase. Since the only difference between H_1 and H_2 lies in the witness used in the proof in the reveal phase, it follows from the robustness w.r.t. $\mathcal{O}$ property of $\langle C, R\rangle$ and the strongly $\mathcal{WI}$ property of the proof that, executions in H_1 and H_2 are indistinguishable.

Hybrid H_3 proceeds the same as H_2 does, except that, S commits to 0^n, instead of the secret r, in Stage 3 of the commit phase. Since the only difference between H_2 and H_3 lies in the value committed to (using $\langle C, R\rangle$) in Stage 3 of the commit phase simulated for A, it follows from the same argument as in H_1 that the executions in H_2 and H_3 are indistinguishable.

Finally, as the view of A in H_3 is emulated perfectly as in the real-execution, we have that the output of Z in H_3 is identical to that in the real-execution. It then follows using a hybrid argument that the ideal and the real executions are indistinguishable in the first case.

Analysis of the second case: In this case, since the simulator S emulates the honest receiver P_j perfectly for A (controlling the committer P_i), the views of A in the ideal and real worlds are identically distributed. Furthermore, we show that the commited value that S extracts from A is almost always the same as the value that A opens to in the reveal phase. Hence the outputs of the honest receiver P_j in the ideal and real woulds are (almost always) identical, and thus so are the outputs of the environment.

Recall that in this case, S sends the value v that A commits to in Stage 2 (obtained from $\mathcal{H}$) as the commit phase input to $\mathcal{F}_{\mathsf{com}}$. Assume for contradiction that v is not the value A opens to later in the reveal stage, with non-negligible probability. Then by the soundness of the proof in the reveal stage, A must have committed to the secret r (i.e., the committed value of Stage 1) in Stage 3, with non-negligible probability. Then we can construct an adversary A' that violates the CCA-security w.r.t. $\mathcal{O}$ of $\langle C, R\rangle$. In the experiment IND_b, The adversary $A'^{\mathcal{O}}$, internally emulates an ideal-execution with A and Z, by emulating the functionality $\mathcal{H}$ using $\mathcal{O}$; it emulates the commit and reveal phases for A as S does, except that, it forwards the commitment of $\langle C, R\rangle$ it receives from the external committer to A as Stage 1—in IND_b, the external commitment is a commitment to value r_b, from $\{r_0, r_1\}$ chosen by A', and uses identity id again chosen by A'; here A' selects r_0 and r_1 at ramdom and sets id to be (P_j, sid)—furthermore, A' also forwards the Stage 3 commitment from A to $\mathcal{O}$ to obtain a decommitment pair (u, d), and outputs u at the end of the execution. Since A' emulates the ideal-execution perfectly for A and Z, the probability that A, in emulation by A', commits to the secret in Stage 3 of the commit phase is identical to that in the ideal-execution. Therefore, by our hypothesis, in IND_0, where the secret is set to r_0, the probability that A commits to r_0 in Stage 3 is non-negligible. However, in IND_1, the probability that A commits to r_0 in Stage 3 is at most $1/2^n$, as the ideal-execution is simulated completely without using r_0. Therefore the outputs of A' in IND_0 and IND_1 are distinguishable. Furthermore, since A' only forwards $\mathcal{O}$ the commitments from A and Z, which use identities of the corrupted parties, A' never queries $\mathcal{O}$ on any commitment that uses the identity of the left interaction (P_j, sid). Hence A' violates the CCA-security w.r.t. $\mathcal{O}$ of $\langle C, R\rangle$.

Analysis of the third case: In this case, S simulates the interaction between two honest parties for A. Since the simulation strategy is the same as that in the first case, it follows from the same proof that the real and ideal executions are indistinguishable in this case.

□

Realizing Any Functionality: First note that to realize any well-formed functionality, it suffices to realize the *ideal oblivious transfer functionality* $\mathcal{F}_{\mathsf{OT}}$ [Rab05, EGL85], which allows a receiver to obtain one out of two bits held by a sender, without revealing to the sender the identity of its selection, as presented in Figure 7. By previous works [Kil92, DBOW88, GMW91, IPS08], this suffices for *unconditionally* implementing any functionality.

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

1. Upon receiving input $(\texttt{Sender}, sid, b_0, b_1)$ from party S, verify that $sid = (S, R, sid')$ for some identity R and $b_0, b_1 \in \{0,1\}$; else ignore the input. Next, record b_0, b_1 and generate a public delayed output $(\texttt{Sender}, sid)$ to R. Ignore further $(\texttt{Sender}, ...)$ inputs.

2. Upon receiving input $(\texttt{Receiver}, sid, i)$ from R, where $i \in \{0,1\}$, wait until a value b_0, b_1 is recorded, and then send a private delayed output $(\texttt{Output}, sid, b_i)$ to R and halt.

3. Upon receiving a message $(\texttt{Corrupt}, sid, P)$ from the adversary, where $P \in \{S, R\}$, send b_0, b_1 to the adversary. Furthermore, if S is corrupted, the adversary now provides values b'_0, b'_1 with each $b'_j \in \{0,1\}$, and no output was yet written to R, then output $(\texttt{Output}, sid, b'_i)$ to R and halt.

Figure 7: The Oblivious Transfer functionality, $\mathcal{F}_{\mathsf{OT}}$

Towards securely realizing $\mathcal{F}_{\mathsf{OT}}$, we would like to rely on the the CLOS-BMR protocol [CLOS02, DBR90]—a *constant-round* protocol that UC-realize $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{com}}$-hybrid model—and simply realize $\mathcal{F}_{\mathsf{com}}$ using our $\mathcal{H}$-EUC secure implementation (described above). If the CLOS-BMR protocol had been a $\mathcal{H}$-EUC realization of $\mathcal{F}_{\mathsf{OT}}$, then the resulting composed protocol would also be $\mathcal{H}$-EUC secure (and we would be done). But recall that UC-security does not necessarily imply $\mathcal{H}$-EUC security (since now the environment is endowed by the super-polynomial oracle $\mathcal{H}$). One way around this problem would be to rely on subexponentially-hard trapdoor permutations in the CLOS-BMR protocol. We take a different route (which dispenses of the extra assumptions): Since the CLOS-BMR protocol is constant-round, we can rely on the *robust* CCA-security property of $\mathcal{O}$ to prove that CLOS-BMR (relying on standard, polynomially-hard, trapdoor permutations) in fact is secure also w.r.t $\mathcal{H}$. More precisely,

Lemma 3. *Assume the existence of enhanced trapdoor permutations. Then, there exists a constant-round $\mathcal{F}_{\mathsf{com}}$-hybrid protocol π_{OT} that $\mathcal{H}$-EUC-emulates $\mathcal{F}_{\mathsf{OT}}$.*

Proof. To prove this lemma, we rely on the previous results from [CLOS02, DBR90][10].

Theorem 6 ([CLOS02, DBR90])**.** *Assume the existence of enhanced trapdoor permutations. Then, there exists a constant-round $\mathcal{F}_{\mathsf{com}}$-hybrid protocol ρ_{OT} that UC-realizes $\mathcal{F}_{\mathsf{OT}}$, with a black-box security proof.*

We show that the protocol ρ_{OT} that UC-emulates $\mathcal{F}_{\mathsf{OT}}$ also $\mathcal{H}$-EUC-emulates $\mathcal{F}_{\mathsf{OT}}$. By Theorem 3, it suffices to show that, for the dummy adversary $\mathcal{D}$, (which simply forwards messages between the honest parties, using ρ_{OT}, and the environment,) there exists a simulator S, such that no environment Z, with access to the helper functionality $\mathcal{H}$, can distinguish its interactions with $\mathcal{D}$ or S. Note that since the dummy adversary $\mathcal{D}$ never accesses $\mathcal{H}$, it is also a valid real-world

[10]Previous results in [CLOS02, DBR90] showed that, assuming the existence of enhanced trapdoor permutations, for every (non-reactive) function g, there exists a constant-round $\mathcal{F}_{\mathsf{com}}$-hybrid protocol that UC-securely evaluates this function. Here we only relies on this result applied to a specific function, that is, the oblivious transfer function.

adversary in the UC-model. Then by the definition of UC-security, there exists a simulator S', such that no environment Z' that obeys the rules of interaction for UC, can distinguish its interaction with $\mathcal{D}$ (and the honest parties using the $\mathcal{F}_{\mathsf{com}}$-hybrid protocol ρ_{OT} in the real-world), from its interaction with S' (and the honest parties using $\mathcal{F}_{\mathsf{OT}}$ in the ideal-world). Then let B_1 be the compound machine that contains $\mathcal{D}$, the honest parties, and $\mathcal{F}_{\mathsf{com}}$ in the real-world, and B_2 the compound machine that contains S', the honest parties, and $\mathcal{F}_{\mathsf{OT}}$ in the ideal-world. We have that:

- no UC-environment Z' can tell apart its interaction with B_1 or B_2, and
- both B_1 and B_2 are constant-round ITMs—since the dummy adversary $\mathcal{D}$ simply forwards messages between the honest parties and the environment, the interaction with B_1 consists of only messages in the protocol ρ_{OT}, and hence has some constant number of rounds k; furthermore, by the indistinguishability of the interaction with B_1 and B_2, the interaction with B_2 also contains only k rounds.

Given the two properties above, we show that S' is also a valid simulator for $\mathcal{D}$ in the $\mathcal{H}$-EUC-model, that is, no environment Z, in the $\mathcal{H}$-EUC-model, can tell apart its interactions with $\mathcal{D}$ (and the honest parties using ρ_{OT}), from that with S' (and honest parties using $\mathcal{F}_{\mathsf{OT}}$). In other words, no environment Z, having access to $\mathcal{H}$, can tell aparts its interaction with B_1 or B_2. Suppose not, and that there is an environment Z that distinguishes interactions with B_1 and B_2. Then by the robustness w.r.t. $\mathcal{O}$ of $\langle C, R\rangle$, and the fact that both B_1 and B_2 are constant-round ITMs, there exists a simulator Z'', such that, the interacion between B_i and $Z^{\mathcal{H}}$, is indistinguishable from that between B_i and Z''. Therefore the environment Z'', without access to $\mathcal{H}$, also distinguishes the interactions with B_1 and B_2. However, this contradicts with the first property above. Hence, we conclude the lemma.

□

Finally, given $\mathcal{F}_{\mathsf{OT}}$, we can securely realize any well-formed functionalities.

Lemma 4. *For every well-formed functionality $\mathcal{F}$, there exists a constant-round $\mathcal{F}_{\mathsf{OT}}$-hybrid protocol ρ that $\mathcal{H}$-EUC-emulates $\mathcal{F}$.*

Proof. This lemma follows essentially from the previous works [Kil92, DBOW88, GMW91, IPS08], which showed that for any well-formed functionality $\mathcal{F}$, there exists a constant-round protocol π that UC-securely realizes $\mathcal{F}$ in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, with a black-box security proof. In fact, it follows syntactically from the same proof as in [Kil92, DBOW88, GMW91, IPS08] that this result holds even for environments that run in sub-exponential time. In particular, for the dummy adversary $\mathcal{D}$ in the $\mathcal{H} - EUC$ model, (who, as argued in the proof of Lemma 3, does not accesses the helper functionality and thus is also a valid UC-adversary), there exists a simulator S in the ideal world, such that no sub-exponential time environment can tell part its interactions with $\mathcal{D}$ or S. Since the helper functionality $\mathcal{H}$ can be implemented in sub-exponential time, we can view environments in the $\mathcal{H}$-EUC model as sub-exponential time machines. Thus, combining Theorem 3, we have that π also $\mathcal{H}$-EUC-emulates $\mathcal{F}$. □

References

[Bea91] Donald Beaver. Foundations of secure interactive computing. In *CRYPTO*, pages 377–391, 1991.

[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 '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 136, Washington, DC, USA, 2001. IEEE Computer Society.

[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.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO '01*, pages 19–40, 2001.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC '00*, pages 235–244, 2000.

[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.

[DBOW88] S. Goldwasser D. Ben-Or and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88*, pages 1–10, 1988.

[DBR90] S. Micali D. Beaver and P. Rogaway. The round complexity of secure protocols. In *STOC '90*, pages 503–513, 1990.

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

[DGS09] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260, 2009.

[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.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GL90] Shafi Goldwasser and Leonid A. Levin. Fair computation of general functions in presence of immoral majority. In *CRYPTO*, pages 77–93, 1990.

[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. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM.

[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.

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

[Gol04] Oded Goldreich. *Foundations of Cryptography — Basic Applications.* Cambridge University Press, 2004.

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

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

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in polyloalgorithm rounds. In *STOC '01*, pages 560–569, 2001.

[Lin04] Yehuda Lindell. Lower bounds for concurrent self composition. In *TCC '04*, pages 203–222, 2004.

[LP09] Huijia Lin and Rafael Pass. Non-malleability amplification. In *STOC '09*, pages 189–198, 2009.

[LPV08] Huijia Lin, Rafael Pass, and Muthuramakrishnan Venkitasubramaniam. Concurrent non-malleable commitments from any one-way function. In *TCC '08*, 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 '09*, pages 179–188, 2009.

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

[MR91] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *CRYPTO*, pages 392–404, 1991.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *EUROCRYPT*, pages 160–176, 2003.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *CRYPTO 2008: Proceedings of the 28th Annual conference on Cryptology*, pages 57–74, Berlin, Heidelberg, 2008. Springer-Verlag.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–413, 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 '02*, 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.

[PV08] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. On constant-round concurrent zero-knowledge. In *TCC '08*, pages 553–570, 2008.

[PW01] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *IEEE Symposium on Security and Privacy*, pages 184–, 2001.

[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 '99*, pages 415–432, 1999.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, pages 433–444, 1991.

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