Concurrently Composable Non-Interactive Secure Computation

Andrew Morgan[1] and Rafael Pass*[2]

[1] Cornell University, Ithaca, USA
asmorgan@cs.cornell.edu
[2] Cornell Tech, New York City, USA
rafael@cs.cornell.edu

Abstract. We consider the feasibility of non-interactive secure two-party computation (NISC) in the plain model satisfying the notion of superpolynomial-time simulation (SPS). While *stand-alone* secure SPS-NISC protocols are known from standard assumptions (Badrinarayanan et al., Asiacrypt 2017), it has remained an open problem to construct a *concurrently composable* SPS-NISC. Prior to our work, the best protocols require 5 rounds (Garg et al., Eurocrypt 2017), or 3 simultaneous-message rounds (Badrinarayanan et al., TCC 2017).

In this work, we demonstrate the first concurrently composable SPS-NISC. Our construction assumes the existence of:

- a non-interactive (weakly) CCA-secure commitment,
- a stand-alone secure SPS-NISC with subexponential security,

and satisfies the notion of "angel-based" UC security (i.e., UC with a superpolynomial-time helper) with perfect correctness.

We additionally demonstrate that both of the primitives we use (albeit only with polynomial security) are necessary for such concurrently composable SPS-NISC with perfect correctness. As such, our work identifies essentially *necessary* and *sufficient* primitives for concurrently composable SPS-NISC with perfect correctness in the plain model.

1 Introduction

Secure two-party computation is a primitive that allows two parties to compute the result $f(x, y)$ of a function f on their respective inputs x,y, while ensuring that nothing else is leaked. In this paper, we focus on secure two-party computation in the setting of minimal communication, where both players send just a *single message*. The first player, called the *receiver*, speaks first, and next the second player, called the *sender*, responds; finally, only the receiver recovers

* Supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, AFOSR Award FA9550-18-1-0267, and a JP Morgan Faculty Award. This material is based upon work supported by DARPA under Agreement No. HR00110C0086. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

the output $f(x, y)$ of the function. Such 2-round protocols are referred to as *non-interactive secure computation protocols (NISC)*.[3]

Security of secure computation protocols is traditionally defined using the *simulation paradigm*, first introduced in [30] and extended in several later works [29,6,40,12]. Roughly speaking, security is defined by requiring that the "view" of any polynomial-time attacker can be simulated by a polynomial-time attacker that participates in an "idealized" version of the protocol where the parties only interact with a trusted party computing f. While this notion of "basic" simulation-based security is often adequate in cases where a protocol is run in isolation, there are several important properties of real-world security that are not considered by this definition. For instance, many protocols interact with other protocols, either through using them as components or sub-protocols or through existing in the same setting; intuitively, it is desirable that a definition of security should provide a guarantee that such a *composition* of multiple provably secure protocols is still secure. Some of the classical definitions of simulation-based security (e.g., [40,12]) in fact did guarantee such a notion of composability.

Concurrently Composable Secure Computation. All of the early definitions of simulation-based security, however, had a caveat; security was only considered when the protocol was executed in a *stand-alone setting* where only a single instance could be executed at a time. Realistically, protocols are often executed in a *concurrent* setting (originally formalized in [20,17,19]) where many instances of a protocol are executed, potentially simultaneously, between many different parties. An adversary in this model may control a large subset of the players, and furthermore is able to observe the results of ongoing interactions in order to adaptively influence future interactions by either reordering communication or changing the behavior of the corrupted parties. Ideally, we would want to be able to show that a protocol is *concurrently secure*, or that a notion analogous to simulation-based security holds even against a more powerful adversary in this multi-instance setting. As with composability, though, concurrent security is not implied by basic definitions of simulation-based security; while definitions such as those of [40,12] guaranteed composable security in a non-concurrent setting, the first definition to achieve both properties was that of *universally composable* (UC) security, first proposed in [13]. At a high level, UC security expands further on the simulation paradigm by considering an "external observer", or *environment*, which runs and observes interactions between an adversary and potentially many concurrent instances of a protocol Π. We say that Π *UC-realizes* some functionality f if the environment cannot distinguish between the "real" interaction and an "ideal" interaction between a polynomial-time simulator and the perfectly secure "idealized" version of the functionality f. Furthermore, if a protocol π UC-realizes some functionality g and Π uses π as a sub-protocol, the composability of UC guarantees that, since the environment cannot distinguish

[3] As is well-known, in this non-interactive setting, it is inherent that only one of the players can receive the output.

interactions with π from simulated interactions with the idealized g, we can effectively replace π with the idealized g when proving Π secure.

While UC security provides extremely strong guarantees, it also has correspondingly restrictive limitations on what can be proven secure. Even in the case of *two-party computation*, impossibility results exist showing that very few functionalities $f(x, y)$ can be computed UC-securely [15]—or, disregarding composability, even concurrently securely [38]—without introducing additional *trusted setup* assumptions.

The notion of *superpolynomial-time simulation* (SPS) [43], a relaxation of UC security which allows the simulator to run in superpolynomial time, has allowed for the construction of several protocols, both for two-party computation [43,47,3,41] and the more general case of *multi-party computation* [5,37,23], which are able to securely realize virtually all functionalities. While some definitions of SPS security provide the same concurrency guarantees as UC security, SPS security fails to uphold many of the desirable composability properties: the problem is that SPS security only requires that any polynomial time attacker can be simulated (in superpolynomial time), but to perform composition, we also need to simulate "simulated attackers", which run in superpolynomial time. The notion of *"angel-based" UC security* [45] and its generalization of *"UC-security with a superpolynomial-time helper"* [16] remedy this issue and provide for a composable notion of concurrent SPS-security: in these models, the simulation is polynomial-time but *both* the adversary and the simulator have access to a "helper" oracle (an "angel") which implements some specific superpolynomial-time functionality. Angel-based security is a strictly stronger notion than SPS security, and it retains all of the composability properties of standard UC security, with the important caveat that composability only holds with protocols that are secure with respect to the same oracle. Furthermore, secure computation protocols are feasible in the angel-based security model [45,39,16,32,33,31]; the most recent constructions have been based on the notion of "CCA-secure" commitments [16], which are commitment schemes that satisfy hiding in the presence of an adversaries that is given access to a "decommitment oracle".

On the Existence of Concurrently-Composable NISC. In this work, we consider the feasibility of concurently composable non-interactive (i.e., 2-round) secure computation protocols, NISCs. As is well known, even if we do not care about concurrency or composability, NISC protocols are not possible *in the plain model* (i.e., without any trusted set-up assumptions) using the standard notion of polynomial-time simulation [28]. On the other hand, if we consider the relaxed notion of SPS security, NISC protocols have been shown to be feasible based on standard assumptions in recent works [43,3,41]. (Indeed, enabling secure 2-round protocols was one of the original motivations behind the notion of SPS security [43].) These works, however, only consider *stand-alone* SPS security.

In fact, even if we require just *concurrent* SPS security (let alone both concurrent and composable), the question of what we can achieved remains open. The state of the art can be summarized as follows:

- [23] proposed the first concurrently secure constant-round protocol based on standard assumptions, and this bound was later reduced to 5 [24].
- [4] presented a three-round concurrently SPS-secure multi-party computation protocol for general functionalities, which can be reduced to two rounds for specific subclasses of functionalities; however, their protocol relies on the *simultaneous-message* model, and so it still requires five (or, for restricted functionalities, three) messages for two-party computation in the standard (synchronous) model.
- Other general two-round concurrently secure multi-party computation protocols (e.g., [7,25,8]) exist which require a common reference string (CRS) as "trusted setup".
- Two recent works (concurrent with and independent from this result) constructed two-round concurrently SPS-secure protocols without trusted setup in the simultaneous-message model (where all participants send a message at the same time in each round). [1] presented a two-round protocol for *two-sided* two-party computation (where both parties receive the output) satisfying concurrent SPS security, and [21] presented a two-round MPC protocol satisfying both concurrent and self-composable SPS security. In contrast to these works, we consider a *synchronous* model where only one participant may send a single message per round (i.e., non-interactive protocols), but we only consider one-sided functionalities.
- For the special case of zero-knowledge arguments of knowledge, [43] presented a 2-round protocol that satisfies concurrent SPS-security; but concurrent security only holds in the setting of "fixed", as opposed to "interchangeable", roles—that is, the attacker can corrupt either all provers, or all verifiers. (On a technical level, this notion of concurrency with "fixed roles" does not deal with *non-malleability* [17].)

Hence, prior work leaves open the question of whether, in the plain model, we can achieve a concurrently secure protocol even for *specific* two-party functionalities, such as zero-knowledge arguments of knowledge, in two *synchronous* (rather than simultaneous-message) rounds.

Meanwhile, for composable "angel-based" security in the plain model, the situation is even worse; the protocol proposed by [16] requires n^ϵ rounds, while [32] reduced this to logarithmic round complexity and [33] further reduced this to a constant. Thus, the literature leaves open the following fundamental problem:

> *Is concurrently composable NISC possible in the plain model, and if so, under what assumptions?*

In fact, we are not aware of NISC protocols even for *specific functionalities* (e.g., zero-knowledge arguments of knowledge) that satisfy *any* "meaningful notion" of concurrent security with "interchangeable roles" (i.e., the adversary can corrupt the sender in some sessions and the receiver in others) even with respect to just *2 concurrent sessions*![4]

[4] In particular, as far as we are aware, even getting a 2-round non-malleble SPS-zero-knowledge argument of knowledge was open.

1.1 Our Results

We solve both of the above questions by demonstrating the existence of a NISC protocol for *general* functionalities satisfying not only concurrent SPS security but also UC security with a superpolynomial-time helper. Our construction relies on the following building blocks:

- A *non-interactive CCA-secure commitment scheme* [42,16,35,9].
- A *stand-alone secure SPS-NISC* with subexponential security [3].

In fact, as we show, a relaxed version of CCA-secure commitments—which we refer to as *weakly CCA-secure commitments*—suffices; this notion differs from the standard notion of CCA security only in that the CCA oracle, given a commitment c, rather than returning both the value v committed to and the randomness r used in the commitment, instead returns just the value v (analogous to the definition of CCA security for encryption schemes [46]). Our main result, then, is as follows:

Theorem 1 (Informal). *Assume there exist a non-interactive weakly CCA-secure commitment scheme and a stand-alone subexponentially SPS-secure NISC protocol for general functionalities. Then there exists a NISC protocol for general functionalities (with perfect correctness) which achieves UC security with a superpolynomial-time helper (i.e., achieves angel-based security).*

We emphasize that before our result, it was not known how to even construct *non-malleable* 2-round protocols in the plain model (i.e., protocols secure under just *two different executions* where the adversary may play different roles) for *any* non-trivial functionality. Furthermore, we demonstrate that the two building blocks we rely on are also *necessary* for concurrently composable SPS-NISC with perfect correctness[5]:

Theorem 2 (Informal). *Assume the existence of a non-interactive NISC for general functionalities (with perfect correctness) satisfying UC security with a superpolynomial-time helper. Then, there exist both a non-interactive weakly CCA secure commitment scheme and a stand-alone secure SPS-NISC for general functionalities.*

Note that the only gap between the assumptions is that our feasibility result (Theorem 1) relies on the existence of a *subexponentially-secure* SPS-NISC, whereas Theorem 2 only shows that a *polynomially-secure* SPS-NISC is needed. But except for this (minor) gap, our work provides a full characterization of the necessary and sufficient primitives for NISC (with perfect correctness) satisfying UC security with a superpolynomial-time helper.

Thus, our work should be interpreted as showing that to upgrade a stand-alone secure NISC to become concurrently composable, the existence of weakly

[5] As usual, perfect correctness means that if both parties act honestly, then the protocol will output the correct result of the computation with probability 1.

CCA-secure commitments is both necessary and sufficient. Our work thus further motivates the importance of studying non-interactive CCA-secure commitments; furthermore, it highlights that perhaps the weaker notion of "weak" CCA security, introduced here, may be more natural than the stronger version used in earlier works.

On the Realizability of the Building Blocks. As just mentioned, our main results demonstrate that the two building blocks—non-interactive weakly CCA-secure commitments and stand-alone SPS-NISC—are both necessary and sufficient for constructing concurrently composable SPS-NISC. SPS-NISC with subexponential security can be constructed based on a variety of standard assumptions, such as subexponential hardness of the Decisional Diffie-Hellman, Quadratic Residuosity, or N^{th} Residuosity assumptions [3] or subexponential hardness of the Learning With Errors assumption [10].

Non-interactive CCA secure commitments, however, require more complex assumptions. They were first constructed in [42] based on adaptive one-way permutations; later, [35] presented such a scheme, albeit with only *unform* security (i.e., security against uniform attackers) based on keyless collision-resistant hash functions, injective one-way functions, non-interactive witness-indistinguish-able arguments (NIWIs), and subexponentially-secure time-lock puzzles. Even more recently, [9] presented a scheme also satisfying non-uniform security by replacing the keyless collision-resistant hash function with a multi-collision-resistant keyless hash function; while their construction is only claimed to achieve "concurrent non-malleability" [44,36] (and not the stronger notion of CCA security), it seems that a relatively minor modification of their analysis (similar to the analysis in [35]) would show that their construction also achieves CCA security when all the underlying primitives satisfy subexponential security.

Overview. We give a technical overview of our main result in Section 2, provide definitions in Section 3, formally state Theorem 1 in Section 4. Due to space limitations, we have deferred the formal proof to the the full version of our paper. In addition, we formalize and prove Theorem 2 in Section 5 (again, missing proofs for this section are provided in the full version.)

2 Technical Overview

In this section, we provide a high-level discussion of our security definition and our protocol. At a high level, UC security expands on the simulation paradigm by considering an "external observer", or *environment*, which runs and observes interactions between an adversary and potentially many concurrent instances of a protocol Π. We say that Π *UC-realizes* some functionality f if the environment cannot distinguish between the "real" interaction and an "ideal" interaction between a polynomial-time simulator and the perfectly secure "idealized" version of the functionality f. We will demonstrate a strong and composable notion of concurrent security using the *externalized UC model* [12,14], where we assume

the adversary, the environment, and the simulator are strictly *polynomial-time* but have access to an "imaginary angel", or a global "helper" entity $\mathcal{H}$ that implements some superpolynomial-time functionality. (This notion was first considered in [45] for the case of non-interactive, stateless, angels) In our case (as in [16]) $\mathcal{H}$ will implement the CCA decommitment oracle $\mathcal{O}$ for a CCA secure commitment; while interacting with a party P, $\mathcal{H}$ will send a valid decommitment in response to any commitments made using that party's identity as the tag. (Since the adversary controls corrupted parties, this effectively means that $\mathcal{H}$ will decommit any commitments with a corrupted party's identifier, but none with an honest party's identifier). CCA security guarantees, then, that an adversary will never be able to break an honest party's commitment; on the other hand, the presence of the helper $\mathcal{H}$ makes it relatively easy for the simulator $\mathcal{S}$ we construct for the definition of UC security to extract information necessary for simulation from corrupted parties' commitments.

Aside from the commitment scheme, our protocol consists of two major subcomponents. First, in order to evaluate the functionality $f(x, y)$, we begin with a NISC protocol that satisfies *stand-alone* security with superpolynomial-time simulation. In order to build this into a protocol satisfying full UC security, however, we will need to leverage the CCA-secure commitment scheme in order to allow the simulator to extract the malicious party's input from their message; since the simulator is restricted to polynomial time (with access to the CCA helper $\mathcal{H}$), this cannot be done by simply leveraging the superpolynomial-time simulator of the underlying NISC. Instead, if both parties commit to their respective inputs and send the commitments alongside the messages of the underlying NISC, the simulator can easily use the CCA helper to extract the inputs from the commitments. This, however, presents another issue: namely, there must be a way to verify that a potentially malicious party commits to the correct input (i.e., the same one they provided to the NISC). For the case of a corrupted sender, this will require the other major component of our protocol: a 2-round zero-knowledge (ZK) interactive argument with SPS security; unsurprisingly, we remark that an appropriate such SPS-ZK protocol can be obtained from an SPS-NISC.

Towards intuitively describing our protocol, we now briefly describe how we deal with extracting from a malicious receiver and sender before presenting the complete protocol.

Dealing with a malicious receiver: using "interactive witness encryption". As suggested above, the first step towards extracting a malicious receiver's input x is to have the receiver commit to their input x and send the commitment c_x with their first-round message. This way, when the receiver is corrupted, the simulator can extract x using the decommitment helper $\mathcal{H}$. Of course, we require a way to verify that the commitment sent by the receiver is indeed a commitment to the correct value of x (i.e., the same as the receiver's input to the NISC which computes $f(x, y)$). We deal with this using a technique reminiscent of the recent non-concurrent NISC protocol of [41], by using the underlying NISC

to implement an "interactive witness encryption scheme".[6] The receiver will, in addition to their input x for f, input the randomness r_x used to generate the commitment c_x, as well as the corresponding decommitment d_x, to the NISC; the sender will input c_x in addition to y, and the NISC will return $f(x, y)$ *if and only if* (c_x, d_x) is a valid commitment of x using randomness r_x. Hence, if the receiver sends an invalid commitment to x to the sender, they receive $\perp$ from the NISC instead of the correct output; otherwise, if it is valid, the simulator can always extract the correct value of x from the commitment using $\mathcal{H}$.

Simulation with a malicious receiver: using a "two-track" functionality. The second key challenge in the corrupted-receiver case is to ensure that we can simulate the sender message of the underlying NISC protocol, since the simulator in this case does not know the sender's input y. To deal with this, we use an SPS-ZK argument to prove that the sender's NISC message is correctly generated, and we additionally add a *two-track* functionality for the underlying NISC and ZK argument to preserve simulatability. First, we add a trapdoor t, chosen at random and committed to by the receiver simultaneously with x. To ensure that the corrupted-receiver simulator can properly simulate the output of the NISC, we "fix" the output when the trapdoor is used; that is, we augment the NISC's functionality yet again to take inputs t' and z^* from the sender and output z^* if the sender provides t' which matches the receiver's trapdoor t. More explicitly, the sender can *program the output* of the computation in case it can recover the trapdoor t selected by the receiver.

The ZK argument will then prove that either (1) there exists a witness w_1 demonstrating that the sender's NISC message is correctly generated (with respect to their input y) given the receiver's first message, OR (2) there exists a witness w_2 which demonstrates that the sender's NISC message was generated using the trapdoor t and no input y (which, in particular, means that the NISC will output $\perp$ if the trapdoor is *incorrect*). The honest sender can provide a witness for statement (1), while the simulator in the malicious receiver case can decommit t using $\mathcal{H}$ to obtain the trapdoor and generate a witness for statement (2).

Dealing with a malicious sender: using an "argument of knowledge". The above, however, is not quite sufficient to simulate for a corrupted sender as well; we furthermore need an *extractability*, or "argument of knowledge", property such that the sender not only proves that there exists such a witness but also demonstrates that it *knows* such a witness—in other words, such a witness should be extractable from the prover's message in superpolynomial time. This will be necessary to show that a corrupted sender cannot provide a valid witness w_2 to the

[6] Recall that *witness encryption* [22] is a primitive where a message m can be encrypted with a statement x so that anyone with a witness w to x can decrypt m, but m cannot be recovered if x is false. Here, we would like c_x to be the "statement" that the commitment is correctly generated, and the randomness r_x and decommitment d_x the "witness".

trapdoor without having recovered the correct trapdoor t and thus broken the security of the commitment scheme.

In our case, since the only extractor available to us is the decommitment oracle $\mathcal{H}$, we implement extractability by using a technique from [43] which adds a *commitment to the witness* to the statement of the proof. The sender provides a witness (w_1, w_2) and two commitments c_1 and c_2, and the proof accepts either if c_1 is a valid commitment to w_1 and w_1 is a valid witness to statement (1) above, or if the respective statement holds for c_2, w_2, and statement (2). This way, a corrupted sender must with overwhelming probability use a witness for statement (1) in its proof (implying that its NISC messages and commitment to y are correctly generated), as, otherwise, a commitment of a correct witness for statement (2) would reveal the trapdoor t when decommitted and thus clearly break CCA security of the commitment scheme. Finally, as w_1 includes y, the commitment c_1 also provides the necessary extractability for the corrupted sender's input y via the decommitment helper $\mathcal{H}$ in the corrupted-sender case.

2.1 Protocol Summary

With the intuition and components described above, we can summarize our full protocol Π for secure two-party computation of a functionality $f(\cdot, \cdot)$:

- The receiver, given input x, generates a random "trapdoor" t and does as follows:
  - Generates commitment c_x for $x||t$ (respectively), using randomness r_x.
  - Generates the first-round message zk_1 of a two-round SPS-ZK argument.
  - Generates the first-round message msg_1 of the underlying NISC protocol π, which will securely compute the functionality h described below, using (x, r_x, t) as its input.

  It sends $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$ to the sender.
- The sender, given input y and the receiver's first-round message $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$, does as follows:
  - Generates the second-round message msg_2 of the underlying NISC π, using $(c_x, y, \perp, \perp)$ as its input and r_{NISC} for randomness.
  - Using witness $w_1 = (r_{\mathsf{NISC}}, y)$ and letting c_1 and c_2 be commitments to w_1 and 0, respectively, generates the second-round message zk_2 of the ZK argument for statement $(\mathsf{msg}_1, \mathsf{msg}_2, c_x, c_1, c_2)$ proving that either:
    (1) there exists a witness $w_1 = (r_{\mathsf{NISC}}, y)$ that demonstrates that msg_2 was correctly and consistently generated with respect to the receiver's first message, the sender's input y, and the randomness r_{NISC}, and c_1 is a valid commitment to w_1, OR:
    (2) there exists a witness $w_2 = (r_{\mathsf{NISC}}, t, z^*)$ that demonstrates that msg_2 was generated using input $(c_x, \perp, t, z^*)$ (i.e., using the trapdoor instead of y), and c_2 is a valid commitment to w_2.

  It sends $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ to the receiver.
- The receiver, given the sender's message $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$, does as follows:

- Verifies that zk_2 is an accepting proof with respect to the statement $(\mathsf{msg}_1, \mathsf{msg}_2, c_x, c_1, c_2)$. Terminates with output $\perp$ if not.
- Evaluates and returns the output z from the NISC π.

The functionality h for the inner NISC, on input (x, r_x, t) from the receiver and (c_x, y, t', z^*) from the sender, does the following:

– Verifies that c_x is correctly generated from $x||t$ and the randomness r_x. Outputs $\perp$ if not.
– If the trapdoor t' given by the sender matches the receiver's trapdoor t, bypasses the computation of f and outputs the sender's input z^*.
– Otherwise, returns $f(x, y)$.

Correctness will follow from correctness of the underlying primitives and the fact that an honest sender and receiver will always generate c_x, msg_2, and zk_2 according to the protocol above; thus, if both parties are honest, the SPS-ZK proof from the sender will always accept and the receiver will always obtain $f(x, y)$ from evaluating GC.

In order to prove that Π $\mathcal{H}$-EUC-securely realizes the ideal two-party computation functionality $\mathcal{T}_f$, we need to prove that, for every polynomial-time environment $\mathcal{Z}$ and adversary $\mathcal{A}$ in the "real" execution of the protocol Π, there exists a polynomial-time simulator $\mathcal{S}$ in the "ideal" execution of the protocol $\Pi(\mathcal{T}_f)$ (where, instead of following the protocol, the receiver and sender send their respective inputs x and y to an instance of $\mathcal{T}_f$ and the receiver gets the output $f(x, y)$) such that $\mathcal{Z}$'s view is indistinguishable between the "real" execution using $\mathcal{A}$ and the "ideal" execution using $\mathcal{S}$. This property needs to hold even when the environment and adversary have access to a superpolynomial-time "helper" $\mathcal{H}$ implementing the CCA decommitment oracle. (Recall from above that the helper will provide a decommitment of any commitment whose tag corresponds to a corrupted party). Below, we provide a high-level sketch of the cases for simulating a corrupted sender and receiver.

2.2 Simulating for a Corrupted Receiver

When the receiver is corrupted, $\mathcal{S}$ first needs to extract the receiver's input x from their first message and send it to the ideal functionality; this is straightforward to do, since both x and the trapdoor t can be retrieved by running the decommitment helper $\mathcal{H}$ on the receiver's input c_x (and the committed values must be the same as the ones given to the NISC in order for the receiver to receive an output). However, $\mathcal{S}$ also needs to simulate the NISC message msg_2, the SPS-ZK proof zk_2, and the commitments c_1 and c_2 to send to the receiver without knowing the corresponding input y.

While one might be tempted to simply use the respective simulators from the definitions of security to simulate the messages for the SPS-ZK argument and the internal NISC, we cannot in fact run either of these simulators inside $\mathcal{S}$, since $\mathcal{S}$ is restricted to (helper-aided) polynomial time whereas, these simulators run in superpolynomial time. So, instead of using the simulators, these messages will

be simulated by running the honest protocols using the trapdoor recovered from c_x. $\mathcal{S}$ can generate the NISC message msg_2 using the input $(c_x, \perp, t, z)$, where z is the output $f(x, y)$ returned from the ideal functionality $\mathcal{T}_f$. In addition, $\mathcal{S}$ can use the second track of the ZK argument with witness $w_2 = (r_{\mathsf{NISC}}, t, z)$, ensuring that it can generate both an accepting proof zk_2 and a NISC message that ensures the correct output ($z = f(x, y)$, contingent on the malicious receiver generating c_x correctly) without knowing the sender's input y.

In a sense, this alternative method of simulating the underlying NISC and ZK argument has interesting parallels to techniques in the context of *obfuscation*, where such two-track approaches are often used to go from indistinguishability-based security to simulation-based security; see e.g. [2,34]. We also note that a technique similar to ours (albeit implemented with garbled circuits rather than a NISC) was used in a very recent work to construct oblivious transfer from new assumptions [18].

Proving that these simulated messages are indistinguishable from the real ones follows through a series of hybrids and relies on complexity leveraging along with the simulation-based security of both primitives. First, in order to switch to the second track of the ZK argument, we need to ensure that the commitment c_2 commits to the trapdoor witness $(r_{\mathsf{NISC}}, t, z)$ rather than to 0. By CCA security of the commitment scheme, commitments of the two values are indistinguishable even by a party (the environment) with access to a decommitment oracle (in this case, the helper $\mathcal{H}$). Notice that, since the sender is honest, $\mathcal{H}$ will not provide the environment with decommitments to commitments generated with the sender's tag, which is precisely the property required of the oracle in the CCA security definition.

Next, we deal with switching to the second track of the SPS-ZK and, respectively, to inputting the trapdoor t to the NISC; we first switch the real proof zk_2 using w_1 to a simulated proof using the simulator for the ZK argument. Next, leveraging the fact that the simulated proof is indistinguishable for any msg_2 satisfying either condition of the ZK language (irrespective of *which* condition) and the fact that the simulator $\mathcal{S}'_R$ for the underlying NISC depends only on the adversary (and not on the specific inputs to the NISC), we can indistinguishably switch from the real NISC message using input $(c_x, y, \perp, \perp)$ to a simulated NISC message using $\mathcal{S}'_R$, and then from there to a real NISC message using the trapdoor input $(c_x, \perp, t, z)$. We then switch the simulated ZK proof back to a real ZK proof, this time using the trapdoor witness w_2; lastly, since the witness w_1 depends on y, we must switch the commitment c_1 for the (now unused) first track of the ZK to commit to 0, which will again follow from CCA security.

Complexity leveraging is required to prove indistinguishability between our hybrids, since we require a NISC secure against adversaries able to run the (superpolynomial-time) simulator of the ZK argument, and in turn a ZK argument secure against adversaries able to internally run the decommitment helper $\mathcal{H}$. Furthermore, while the intermediate hybrids clearly run in superpolynomial time, we note that the final simulator $\mathcal{S}$ will still run in polynomial time (with $\mathcal{H}$) and is hence still sufficient to prove the notion of "angel-based" UC security.

To summarize, the corrupted-receiver simulator $\mathcal{S}_R$ proceeds as follows:

- Receives the receiver's first-round message $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$.
- Uses the helper $\mathcal{H}$ to decommit c_x, receiving x^* and t.
- Sends x^* to the ideal functionality $\mathcal{T}_f$ and receives the output z.
- Generates the second-round message msg_2 of the underlying NISC π, using $(c_x, \perp, t, z)$ as its input and r_{NISC} for randomness.
- Using witness $w_2 = (r_{\mathsf{NISC}}, t, z)$ and letting c_1 and c_2 be commitments to 0 and w_2, respectively, generates the second-round message zk_2 of the ZK argument for the language described above and statement $(\mathsf{msg}_1, \mathsf{msg}_2, c_x, c_1, c_2)$.
- Sends $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ to the receiver.

2.3 Simulating for a Corrupted Sender

When the sender is corrupted, $\mathcal{S}$ first needs to simulate the receiver's message $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$ to send to the sender; then, on receiving the sender's message $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$, $\mathcal{S}$ needs to either output $\perp$ (if the sender's message does not verify) or extract the sender's input y to send to the ideal functionality so that the honest receiver gets the correct output $f(x, y)$.

Simulating the first message without knowledge of x will require two changes: making c_x commit to $0||t$ rather than to $x||t$, and respectively changing the first NISC message to use 0 in place of the input x (since, as before, we cannot use a simulated NISC message due to simulation being superpolynomial-time).

We show indistinguishability through a series of hybrids similar to the corrupted receiver case. First, we can use simulation-based security to switch the real NISC message (with input x) to a simulated NISC message using the simulator $\mathcal{S}'_S$ for π. Next, the first message no longer depends on x, so we can leverage CCA security to indistinguishably switch c_x to commit to 0 instead. A minor subtlety with this step is that the polynomial-time adversary for CCA security cannot run the superpolynomial-time simulator $\mathcal{S}'_S$, so instead we leverage non-uniformity and provide the simulated first message of the NISC to the CCA security adversary as non-uniform advice. Finally, we can again leverage simulation-based security (and the input-independence of the simulator $\mathcal{S}'_S$) to switch from the simulated message to another real message using the input 0.

It remains to consider the receiver's output; the honest receiver will output the result from the ideal functionality in the ideal experiment, but we need to ensure that the receiver correctly outputs $\perp$ when the malicious sender provides invalid inputs in its second-round message. On receiving the sender's message $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$, the simulator will extract the malicious party's input by using the helper $\mathcal{H}$ to decommit c_1 (a commitment to the witness w_1, which contains y) and then verify the sender's message. If verification is successful, $\mathcal{S}$ will send the resulting value y^* to the ideal functionality (which will return the result to the honest receiver); if not, it will terminate with output $\perp$.

By soundness of the ZK argument, if $\mathcal{S}$ does not output $\perp$, then the sender is overwhelmingly likely to have provided a proof in zk_2 corresponding to a valid witness; furthermore, we can assert that this witness is overwhelmingly likely

to be a witness $w_1 = (r_{\mathsf{NISC}}, y)$ to part (1) of the ZK argument, since, if the sender could figure out an accepting witness $w_2 = (r_{\mathsf{NISC}}, t, z^*)$ for part (2) with non-negligible probability, this would imply that an adversary could recover this by running a decommitment oracle on the commitment c_2 and subsequently use it to break CCA security of the commitment c_x (which contains t) sent by the receiver in the first round.[7]

Given a valid witness to part (1), then, it must be the case that c_1 is a valid commitment to w_1 and that msg_2 is correctly generated with respect to the y given in w_1—so, on inputs corresponding to a valid commitment c_x of $x||t$, the internal NISC π will output $f(x, y)$ for the same y the simulator receives by decommitting c_1. Hence, we can simulate the output by, if verification passes, having the receiver return the output from the ideal functionality (exactly as in the ideal interaction), which will always be $f(x, y)$ given the y extracted from c_1; the above argument shows that this strategy will produce an output identical to that of the internal NISC with overwhelming probability. Notably, this simulated output is now independent of the value of x used to generate the first-round message (and instead relies on the x sent to the ideal functionality by the honest receiver).

This gives us the completed corrupted-sender simulator $\mathcal{S}_S$, which proceeds as follows:

- Generates a random "trapdoor" t.
- Generates commitment c_x for $0||t$ (respectively), using randomness r_x.
- Generates the first-round message zk_1 of a two-round ZK argument.
- Generates the first-round message msg_1 of the underlying NISC protocol π, which will securely compute the functionality h described below, using $(0, r_x, t)$ as its input.
- Sends $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$ to the sender.

- Receives the sender's message $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$.
- Verifies that zk_2 is an accepting proof with respect to the statement $(\mathsf{msg}_1, \mathsf{msg}_2, c_x, c_1, c_2)$. Terminates with output $\perp$ if not.
- Uses the helper $\mathcal{H}$ to recover w_1 (including y^*) from the commitment c_1.
- Verifies that w_1 is a valid witness for the statement $(\mathsf{msg}_1, \mathsf{msg}_2, c_x, c_1, c_2)$. If not, returns $\perp$.
- Sends y^* to the ideal functionality $\mathcal{T}_f$, which will return the output $f(x, y^*)$ to the receiver.

3 Definitions

3.1 Non-Interactive Secure Computation

We start by defining *non-interactive secure computation* (NISC).

[7] In particular, notice that the commitments c_2 and c_x are generated by different parties and hence *using different tags*—hence, an adversary breaking CCA security with respect to c_x's tag is allowed to decommit c_2.

Definition 1 ([41], based on [48,27,3]). *A **non-interactive two-party computation protocol** for computing some functionality $f(\cdot,\cdot)$ (where f is computable by a polynomial-time Turing machine) is given by three PPT algorithms $(\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ defining an interaction between a sender S and a receiver R, where only R will receive the final output. The protocol will have common input 1^n (the security parameter); the receiver R will have input x, and the sender will have input y. The algorithms $(\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ are such that:*

- $(\mathsf{msg}_1, \sigma) \leftarrow \mathsf{NISC}_1(1^n, x)$ *generates R's message msg_1 and persistent state σ (which is not sent to S) given the security parameter n and R's input x.*
- $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, y)$ *generates S's message msg_2 given S's input y and R's message msg_1.*
- $out \leftarrow \mathsf{NISC}_3(\sigma, \mathsf{msg}_2)$ *generates R's output out given the state σ and S's message msg_2.*

We restrict our attentions to protocols satisfying perfect correctness*:*

- ***Correctness.*** *For any parameter $n \in \mathbb{N}$ and inputs x, y:*

$$Pr[(\mathsf{msg}_1, \sigma) \leftarrow \mathsf{NISC}_1(1^n, x) : \mathsf{NISC}_3(\sigma, \mathsf{NISC}_2(\mathsf{msg}_1, y)) = f(x, y)] = 1$$

Externalized Universally Composable Security. To define the notion of security proven in our main theorem, we use the framework of *universally composable security* [12,13], extended to include access to superpolynomial "helper functionalities" [14,16]. Specifically, we prove UC security in the presence of an external helper which allows the adversary to break the commitments of corrupted parties.

Model of execution. We recall the discussion of UC security with external helper functionalities provided in [16]. Consider parties represented by polynomial-time interactive Turing machines [30]; the model contains a number of parties running instances of the protocol Π, as well as an *adversary* $\mathcal{A}$ and an *environment* $\mathcal{Z}$. The environment begins by invoking the adversary on an arbitrary input, and afterwards can proceed by invoking parties which participate in single instances of the protocol Π by providing them with their respective inputs, as well as a *session identifier* (which is unique for each instance of the protocol Π) and a *party identifier* (which is unique among the participants in each session). The environment can furthermore read the output of any party involved in some execution of Π, as well as any output provided by the adversary.

For the purposes of UC security, we will restrict our attention to environments which may only invoke a single session of the protocol Π—that is, any instances invoked must have the same session identifier. Concurrent and composable security (i.e., against more generalized environments) will follow from this via a *universal composition theorem*, which we will state later in this section.

The adversary, on the other hand, is able to control all communication between the various parties involved in executions of Π, and to furthermore modify the outputs of certain *corrupted* parties (which we here assume are decided

non-adaptively, i.e., every party is either invoked as permanently corrupted or permanently uncorrupted). Uncorrupted parties will always act according to the protocol Π, and we assume that the adversary only delivers messages from uncorrupted parties that were actually intended to be sent (i.e., authenticated communication); the adversary can, on the other hand, deliver any message on behalf of a corrupted party. The adversary can also send messages to and receive them from the environment at any point.

We will furthermore assume a notion of security using an "imaginary angel" [45], which can be formalized in the *externalized UC* (EUC) setting [14]; both the corrupted parties and environment will have access to an external *helper functionality* $\mathcal{H}$, also defined as an interactive Turing machine—unlike the participants, adversary, or environment, however, $\mathcal{H}$ is not restricted to polynomial running time. $\mathcal{H}$ is persistent throughout the execution and is invoked by the environment immediately after the adversary is; furthermore, $\mathcal{H}$ must be immediately informed of the identity of all corrupted parties when parties are determined by the environment to be corrupted.

Finally, while honest players can only be invoked on a single session identifier, we allow the adversary to invoke $\mathcal{H}$ on behalf of corrupt parties using potentially *arbitrary* session identifiers; this is needed to prove the composition theorem.

The execution ends when the environment halts, and we assume the output to be the output of the environment. We let $\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(1^n, z)$ denote the distribution of the environment's output, taken over the random tape given to $\mathcal{A}$, $\mathcal{Z}$, and all participants, in the execution above (with a single session of Π), where the environment originally gets as input security parameter 1^n and auxiliary input z. We say that Π *securely emulates* some other protocol Π' if, for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that the environment $\mathcal{Z}$ is unable to tell the difference between the execution of Π with $\mathcal{A}$ and the execution of Π' with $\mathcal{S}$—that is, intuitively, the environment gains the same information in each of the two executions. Formally:

Definition 2 (based on [16]). *For some (superpolynomial-time) interactive Turing machine $\mathcal{H}$, we say a protocol Π **$\mathcal{H}$-EUC-emulates** some protocol Π' if, for any polynomial-time adversary $\mathcal{A}$, there exists some simulated polynomial-time adversary $\mathcal{S}$ such that, for any non-uniform polynomial-time environment $\mathcal{Z}$ and polynomial-time distinguisher D, there exists negligible $\nu(\cdot)$ such that, for any $n \in \mathbb{N}$ and $z \in \{0,1\}^*$:*

$$|Pr[D(\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(1^n, z)) = 1] - Pr[D(\mathsf{Exec}_{\Pi',\mathcal{S},\mathcal{Z}}(1^n, z)) = 1]\,| \leq \nu(n)$$

To prove that a protocol Π *securely realizes* an ideal functionality $\mathcal{T}$, we wish to show that it securely emulates an "ideal" protocol $\Pi(\mathcal{T})$ in which all parties send their respective inputs to an instance of $\mathcal{T}$ with the same session identifier and receive the respective output; note that the adversary does not receive the messages to or from each instance of $\mathcal{T}$.

Definition 3 (based on [16]). *For some (superpolynomial-time) interactive Turing machine $\mathcal{H}$, we say a protocol Π **$\mathcal{H}$-EUC-realizes** some functionality $\mathcal{T}$ if it $\mathcal{H}$-EUC-emulates the protocol $\Pi(\mathcal{T})$ given above.*

In the case of two-party computation for functionality f, $\mathcal{T}$ will simply receive inputs x from the receiver and y from the sender and return $f(x, y)$ to the receiver:

Definition 4. *For some (superpolynomial-time) interactive Turing machine $\mathcal{H}$, we refer to a non-interactive two-party computation protocol Π for some functionality $f(\cdot, \cdot)$ as **$\mathcal{H}$-EUC-secure** if it $\mathcal{H}$-EUC-realizes the functionality $\mathcal{T}_f$, which, on input x from a receiver R and input y from a sender S, returns $f(x, y)$ to R.*

Remarks. Notice that, since $\mathcal{Z}$'s output is a (randomized) function of its view, it suffices to show that $\mathcal{Z}$'s view cannot be distinguished by any polynomial-time distinguisher D between the respective experiments. We can also without loss of generality assume that the environment $\mathcal{Z}$ in the real execution effectively runs the adversary $\mathcal{A}$ internally and forwards all of $\mathcal{A}$'s messages to and from other parties by using a "dummy adversary" $\mathcal{D}$ which simply forwards communication from $\mathcal{Z}$ to the respective party. This allows us to effectively view the environment $\mathcal{Z}$ and adversary $\mathcal{A}$ as a single entity.

Furthermore, observe that we use a *polynomial-time* simulator $\mathcal{S}$ in our definition of security. [28] shows that two-round secure computation protocols cannot be proven secure with standard polynomial-time simulation; hence, many protocols are proven secure using superpolynomial-time simulators (a technique originally proposed by [43,45]). Indeed, we note that, if $\mathcal{H}$ runs in time $T(\cdot)$, then a protocol that $\mathcal{H}$-EUC-realizes some functionality $\mathcal{T}$ with polynomial-time simulation will also UC-realize $\mathcal{T}$ with $\mathsf{poly}(T(\cdot))$-time simulation; hence, in a way, the simulator $\mathcal{S}$ we propose in our security definition can still be considered to do a superpolynomial-time amount of "work".

Universal composition. The chief advantage of the UC security paradigm is the notion of *universal composition*; intuitively, if a protocol ρ UC-realizes (or, respectively, $\mathcal{H}$-EUC-realizes) an ideal functionality $\mathcal{T}$, then it is "composable" in the sense that any protocol that uses the functionality $\mathcal{T}$ as a primitive derives the same security guarantees from the protocol ρ as they would the ideal functionality.

More formally, given an ideal functionality $\mathcal{T}$, let us define a *$\mathcal{T}$-hybrid protocol* as one where the participating parties have access to an unbounded number of copies of the functionality $\mathcal{T}$ and may communicate directly with these copies as in an "ideal" execution (i.e., without communication being intercepted by the adversary). Each copy of $\mathcal{T}$ will have a unique session identifier, and their inputs and outputs are required to contain the respective identifier.

Then, if Π is a $\mathcal{T}$-hybrid protocol, and ρ is a protocol which realizes $\mathcal{T}$, then we can define a *composed protocol* Π^ρ by modifying Π so that the first message sent to $\mathcal{T}$ is instead an invocation of a new instance of ρ with the same session identifier and the respective message as input, and so that further messages are likewise relayed to the same instance of ρ instead, again with their contents as the respective input. Any output from an instance of ρ is substituted for the

respective output of the corresponding instance of $\mathcal{T}$. The following powerful theorem, then, states the notion of composability intuitively described above.

Theorem 3 (Relativized Universal Composition [12,16]). *For some ideal functionality $\mathcal{T}$ and helper functionality $\mathcal{H}$, if Π is a $\mathcal{T}$-hybrid protocol, and ρ is a protocol that $\mathcal{H}$-EUC-realizes $\mathcal{T}$, then Π^ρ $\mathcal{H}$-EUC-emulates Π.*

Stand-alone Security. As one of the key building blocks of our UC-secure protocol, we use a non-interactive secure computation protocol which satisfies the strictly weaker notion of *stand-alone security with superpolynomial-time simulation*. We recall the definition (as given in [41]) below:

- Consider a *real* experiment defined by an interaction between a sender S with input y and a receiver R with input x as follows:
  - R computes $(\mathsf{msg}_1, \sigma) \leftarrow \mathsf{NISC}_1(1^n, x)$, stores σ, and sends msg_1 to S.
  - S, on receiving msg_1, computes $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, y)$ and sends msg_2 to R.
  - R, on receiving msg_2 computes $out \leftarrow \mathsf{NISC}_3(\sigma, \mathsf{msg}_2)$ and outputs out.

  In this interaction, one party $I \in \{S, R\}$ is defined as the *corrupted* party; we additionally define an *adversary*, or a polynomial-time machine $\mathcal{A}$, which receives the security parameter 1^n, an auxiliary input z, and the inputs of the corrupted party I, and sends messages (which it may determine arbitrarily) in place of I.
  Letting Π denote the protocol to be proven secure, we shall denote by $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ the random variable, taken over all randomness used by the honest party and the adversary, whose output is given by the outputs of the honest receiver (if $I = S$) and the adversary (which may output an arbitrary function of its view).
- Consider also an *ideal* experiment defined by an interaction between a sender S, a receiver R, and a *trusted party* $\mathcal{T}_f$, as follows:
  - R sends x to $\mathcal{T}_f$, and S sends y to $\mathcal{T}_f$.
  - $\mathcal{T}_f$, on receiving x and y, computes $out = f(x, y)$ and returns it to R.
  - R, on receiving out, outputs it.

  As with the real experiment, we say that one party $I \in \{S, R\}$ is corrupted in that, as before, their behavior is controlled by an adversary $\mathcal{A}$. We shall denote by $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{A},I}(1^n, x, y, z)$ the random variable, once again taken over all randomness used by the honest party and the adversary, whose output is again given by the outputs of the honest receiver (if $I = S$) and the adversary.

Definition 5 ([41], based on [48,27,43,45,3]). *Given a function $T(\cdot)$, a non-interactive two-party protocol $\Pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ between a sender S and a receiver R, and functionality $f(\cdot, \cdot)$ computable by a polynomial-time Turing machine, we say that Π* ***securely computes f with $T(\cdot)$-time simulation,*** *or that Π is a* ***non-interactive (stand-alone) secure computation protocol***

(with $T(\cdot)$-time simulation) *for computing f, if Π is a non-interactive two-party computation protocol for computing f and, for any polynomial-time adversary $\mathcal{A}$ corrupting party $I \in \{S, R\}$, there exists a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}$ such that, for any $T(n) \cdot \mathsf{poly}(n)$-time algorithm $D : \{0,1\}^* \to \{0,1\}$, there exists negligible $\epsilon(\cdot)$ such that for any $n \in \mathbb{N}$ and any inputs $x, y \in \{0,1\}^n, z \in \{0,1\}^*$, we have:*

$$\left| Pr[D(\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)) = 1] - Pr\left[D(\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)) = 1\right]\right| < \epsilon(n)$$

where the experiments and distributions Out *are as defined above.*

Furthermore, if Π securely computes f with $T(\cdot)$-time simulation for $T(n) = n^{\log^c(n)}$ for some constant c, we say that Π is ***stand-alone secure with quasi-polynomial simulation****.*

Badrinarayanan et al. [3] demonstrates that stand-alone secure NISC protocols with quasi-polynomial simulation exist assuming the existence of a notion of "weak OT", which in turn can be based on subexponential versions of standard assumptions [3,10]:

Theorem 4 ([3,10]). *Assuming subexponential hardness of any one of the Decisional Diffie-Hellman, Quadratic Residuosity, N^{th} Residuosity, or Learning With Errors assumptions, then for any constants $c < c'$ and any polynomial-time Turing-computable functionality $f(\cdot,\cdot)$ there exists a (subexponentially)* stand-alone secure *non-interactive two-party computation protocol with $T(\cdot)$-time security and $T'(\cdot)$-time simulation for $T(n) = n^{log^c(n)}$ and $T'(n) = n^{log^{c'}(n)}$.*

3.2 SPS-ZK Arguments

We proceed to recalling the definition of interactive arguments.

Definition 6 ([11,30,26]). *We refer to an interactive protocol (P, V) between a probabilistic* prover *P and a* verifier *V as an* ***interactive argument*** *for some language $\mathcal{L} \subseteq \{0,1\}^*$ if the following conditions hold:*

1. ***Completeness.*** *There exists a negligible function $\nu(\cdot)$ such that, for any $x \in \mathcal{L}$:*
$$Pr[\langle P, V\rangle(x) = \mathsf{Accept}] \geq 1 - \nu(|x|)$$
2. ***$T(\cdot)$-time soundness.*** *For* any *non-uniform probabilistic $T(\cdot)$-time prover P^* (not necessarily honest), there exists a negligible function $\nu(\cdot)$ such that, for any $x \notin \mathcal{L}$:*
$$Pr[\langle P^*, V\rangle(x) = \mathsf{Accept}] \leq \nu(|x|)$$
Furthermore, if the above holds even if the statement $x \notin \mathcal{L}$ can be adaptively chosen by the cheating prover anytime prior to sending its last message, we call such a protocol ($T(\cdot)$-time) ***adaptively sound****.*

We also require a notion of *zero-knowledge* [30] with superpolynomial simulation (SPS-ZK) [43], which states that the prover's witness w should be "hidden" from the verifier in the sense that proofs of a particular statement $x \in L$ should be simulatable in a manner independent of w:

Definition 7 ([43]). *We refer to an interactive argument for some NP language L (with witness relation R_L) as $T'(\cdot)$-**time simulatable zero-knowledge with $T(\cdot)$-time security** (or $(T(\cdot), T'(\cdot))$-simulatable zero-knowledge) if, for any $T(\cdot)$-time cheating verifier V^* (which can output an arbitrary function of its view), there exists a $T'(\cdot)$-time simulator Sim and negligible function $\nu(\cdot)$ such that, for any $T(\cdot)$-time non-uniform distinguisher D, given any statement $x \in L$, any witness $w \in R_L(x)$, and any auxiliary input $z \in \{0,1\}^*$, it holds that:*

$$|Pr[D(x, \langle P(w), V^*(z)\rangle(x)) = 1] - Pr[D(x, \mathsf{Sim}(x, z)) = 1]\,| \leq \nu(|x|)$$

Our construction will use a two-round adaptively sound zero-knowledge argument consisting of three polynomial-time algorithms, $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$, defining the following interaction $\langle P, V\rangle$:

- V runs $(\mathsf{zk}_1, \sigma) \leftarrow \mathsf{ZK}_1(1^n)$, which takes as input the security parameter n and generates a first message zk_1 and persistent state σ.
- P runs $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{wi}_1, x, w)$, which takes as input the first message wi_1, a statement x, and a witness w, and returns a second message zk_2.
- V runs $\{\mathsf{Accept}, \mathsf{Reject}\} \leftarrow \mathsf{ZK}_3(\mathsf{zk}_2, x, \sigma)$, which takes as input a second message zk_2, a statement x, and the persistent state σ, and returns Accept if zk_2 contains an accepting proof that $x \in L$ and Reject otherwise.

We observe that, in fact, this primitive is implied by the existence of a *standalone secure NISC* (see Definition 5).

Theorem 5. *For any constants $c < c'$, letting subexponential functions $T(n) = n^{log^c(n)}$ and $T'(n) = n^{log^{c'}(n)}$, then, if there exists a subexponentially standalone secure non-interactive two-party computation protocol for any polynomial-time Turing-computable functionality $f(\cdot,\cdot)$ with $T(\cdot)$-time security and $T'(\cdot)$-time simulation, then there exists a two-round interactive argument with $T(\cdot)$-time adaptive soundness and $(T(\cdot), T'(\cdot))$-simulatable zero-knowledge.*

The construction and its proof of security is straightforward, but for completeness we provide it in the full version.

3.3 Non-Interactive CCA-secure Commitments

Our construction will rely on non-interactive (single-message) tag-based commitment schemes satisfying the notion of CCA security [42,16,35].

Definition 8 (based on [35]). *A **non-interactive tag-based commitment scheme** (with $t(\cdot)$-bit tags) consists of a pair of polynomial-time algorithms $(\mathsf{Com}, \mathsf{Open})$ such that:*

– $c \leftarrow \mathsf{Com}(1^n, \mathsf{id}, v; r)$ *(alternately denoted* $\mathsf{Com}_{\mathsf{id}}(1^n, v; r)$*) takes as input an identifier (tag)* $\mathsf{id} \in \{0,1\}^{t(n)}$*, a value* v*, randomness* r*, and a security parameter* n*, and outputs a commitment* c*. We assume without loss of generality that the commitment* c *includes the respective tag* id*.*
– $\{\mathsf{Accept}, \mathsf{Reject}\} \leftarrow \mathsf{Open}(c, v, r)$ *takes as input a commitment* c*, a value* v*, and randomness* r*, and returns either* Accept *(if* c *is a valid commitment for* v *under randomness* r*) or* Reject *(if not).*

We consider commitment schemes having the following properties:

1. ***Correctness:*** *For any security parameter* $n \in \mathbb{N}$*, any* $v, r \in \{0,1\}^*$*, and any* $\mathsf{id} \in \{0,1\}^{t(n)}$*:*

$$\Pr[c \leftarrow \mathsf{Com}(1^n, \mathsf{id}, v; r) : \mathsf{Open}(c, v, r) = \mathsf{Accept}] = 1$$

2. ***Perfect binding (for sufficiently large inputs):*** *There exists* $n \in \mathbb{N}$ *such that, for any commitment string* c*, values* v, v' *with* $|v| \geq n$ *or* $|v'| \geq n$*, and randomness* r, r'*, if it is true that* $\mathsf{Open}(c, v, r) = \mathsf{Accept}$ *and* $\mathsf{Open}(c, v', r') = \mathsf{Accept}$*, then* $v = v'$*.*[8]
3. $T(\cdot)$***-time hiding:*** *For any* $T(\cdot)$*-time non-uniform distinguisher* D *and fixed polynomial* $p(\cdot)$*, there exists a negligible function* $\nu(\cdot)$ *such that, for any* $n \in \mathbb{N}$*, any* $\mathsf{id} \in \{0,1\}^{t(n)}$ *and any values* $v, v' \in \{0,1\}^{p(n)}$*:*

$$|\Pr[D(\mathsf{Com}(1^n, \mathsf{id}, v)) = 1] - \Pr[D(\mathsf{Com}(1^n, \mathsf{id}, v')) = 1]| \leq \nu(n)$$

For our construction, we require a strictly stronger property than just hiding: hiding should hold even against an adversary with access to a "decommitment oracle". This property is known as *CCA security* due to its similarity to the analogous notion for encryption schemes [46]. We introduce a weakening of CCA security, to which we shall refer as "weak CCA security", which is nonetheless sufficient for our proof of security, and, as we shall prove in Section 5, is *necessary* for our proof of security as well. We define this as follows:

Definition 9. *Let* $\mathcal{O}^*$ *be an oracle which, given a commitment* c*, returns a valid committed value* v*—that is, such that there exists some randomness* r *for which* $\mathsf{Open}(c, v, r) = \mathsf{Accept}$*.*

A tag-based commitment scheme $(\mathsf{Com}, \mathsf{Open})$ *is* $T(\cdot)$***-time weakly CCA-secure*** *with respect to* $\mathcal{O}^*$ *if, for any polynomial-time adversary* $\mathcal{A}$*, letting* $\mathsf{Exp}_b(\mathcal{O}^*, \mathcal{A}, n, z)$ *(for* $b \in \{0,1\}$*) denote* $\mathcal{A}$*'s output in the following interactive experiment:*

– $\mathcal{A}$*, on input* $(1^n, z)$*, is given oracle access to* $\mathcal{O}^*$*, and adaptively chooses values* v_0, v_1 *and tag* id*.*
– $\mathcal{A}$ *receives* $\mathsf{Com}(1^n, \mathsf{id}, v_b)$ *and returns an arbitrary output; however,* $\mathcal{A}$*'s output is replaced with* $\perp$ *if* $\mathcal{O}^*$ *was ever queried on any commitment* c *with tag* id*.*

[8] We remark that this property is stronger than statistical binding but weaker than fully perfect binding.

then, for any $T(\cdot)$-time distinguisher D, there exists negligible $\nu(\cdot)$ such that, for any $n \in \mathbb{N}$ and any $z \in \{0,1\}^*$, it holds that:

$$|\Pr[D(\mathsf{Exp}_0(\mathcal{O}^*, \mathcal{A}, n, z)) = 1] - \Pr[D(\mathsf{Exp}_1(\mathcal{O}^*, \mathcal{A}, n, z)) = 1]| \leq \nu(n)$$

We remark that the only difference from the "standard" notion of CCA security is that the CCA oracle, given a commitment c, rather than returning both the value v committed to and the randomness r used in the commitment, instead returns just the value v. This is similar to the definition of CCA security commonly used for encryption schemes [46].

4 Results

We state our main theorem and the respective protocol in this section.

Input: A commitment c, which without loss of generality contains identity id and was sent by party P in session S.
Output: A value v or the special symbol $\perp$.

Functionality:

1. Verify that $\mathsf{id} = (S, P)$ and return $\perp$ if not.
2. Otherwise, run the oracle $\mathcal{O}$ (from the definition of weak CCA security) to find a valid decommitment v (i.e., such that, for some randomness r $\mathsf{Open}(c, v, r) = \mathsf{Accept}$), and return it, or return $\perp$ if there is no valid decommitment (i.e., $\mathcal{O}$ returns $\perp$).

Fig. 1: Decommitment helper $\mathcal{H}$ for a weakly CCA-secure commitment scheme $(\mathsf{Com}, \mathsf{Open})$.)

Theorem 6. *If there exist superpolynomial-time functions $T_{\mathsf{Com}}(\cdot) = n^{\log^{c_0}(n)}$, $T_{\mathsf{ZK}}(\cdot) = n^{\log^{c_1}(n)}$, $T_{\mathsf{Sim}}(\cdot) = n^{\log^{c_2}(n)}$, and $T_\pi(\cdot) = n^{\log^{c_3}(n)}$ for constants $0 < c_0 < c_1 < c_2 < c_3$ so that there exist (1) a non-interactive weakly CCA-secure commitment scheme with respect to a $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$, (2) a non-interactive computation protocol for general polynomial-time Turing-computable functionalities satisfying $T_{\mathsf{ZK}}(\cdot)$-time stand-alone security and $T_{\mathsf{Sim}}(\cdot)$-time simulation, and (3) a non-interactive computation protocol for general polynomial-time Turing-computable functionalities satisfying $T_\pi(\cdot)$-time stand-alone security (and $T'(\cdot)$-time simulation for some $T'(\cdot) \gg T_\pi(\cdot)$), then, for any polynomial-time Turing-computable functionality $f(\cdot,\cdot)$, the protocol Π given in Figure 2 for computing f is an $\mathcal{H}$-EUC-secure non-interactive secure computation protocol with respect to the helper $\mathcal{H}$ in Figure 1.*

Let $T_{\mathsf{Com}}(\cdot), T_{\mathsf{ZK}}(\cdot), T_{\mathsf{Sim}}(\cdot), T_{\pi}(\cdot)$ be as given in the theorem. Π will use the following primitives:

- $(\mathsf{Com}, \mathsf{Open})$, a secure commitment scheme satisfying weak CCA security with respect to some oracle $\mathcal{O}$ having running time $T_{\mathsf{Com}}(n)$. This is primitive (1) given in the theorem.
- $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$, a two-message interactive argument which satisfies $T_{\mathsf{ZK}}(n)$-time adaptive soundness and $(T_{\mathsf{ZK}}(\cdot), T_{\mathsf{Sim}}(\cdot))$-simulatable zero-knowledge (with respective $T_{\mathsf{Sim}}(\cdot)$-time simulator $\mathsf{Sim}_{\mathsf{ZK}}$). By Theorem 5, this can be constructed from the primitive (2) given in the theorem.
- $\pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$, a *stand-alone secure* non-interactive two-party computation protocol for the functionality h given in Figure 3 satisfying $T_{\pi}(\cdot)$-time security and $T'(\cdot)$-time simulation for some $T'(n) \gg T_{\pi}(n)$. This is implied by primitive (3) in the theorem.

We provide the complete proof, which constructs the polynomial-time simulator $\mathcal{S}$ (aided by $\mathcal{H}$) required for the definition of $\mathcal{H}$-EUC-security, in the full version.

5 Minimality of Assumptions

In this section, we prove that the protocol we construct in Theorem 6 can be constructed using nearly minimal assumptions—that is, that a NISC protocol satisfying externalized UC security implies both a (polynomial-time) stand-alone secure NISC protocol with superpolynomial-time simulation and weakly CCA-secure commitments. Thus, these primitives are not only sufficient but also *necessary* for the existence of an externalized UC-secure NISC. The only gap between the sufficient and necessary conditions is that Theorem 6 requires a stand-alone NISC having simulation-based security with respect to subexponential-time distinguishers, whereas one can only construct a polynomial-time secure stand-alone NISC from our definition of UC security.

Theorem 7. *Assume the existence of a protocol $\Pi = (\pi_1, \pi_2, \pi_3)$ for non-interactive computation of any polynomial-time Turing-computable functionality $f(\cdot, \cdot)$; further assume that Π satisfies the notion of UC security with respect to some superpolynomial-time helper $\mathcal{H}$. Then there exist both a stand-alone secure non-interactive two-party computation protocol (for any polynomial-time Turing-computable functionality $h(\cdot, \cdot)$) with superpolynomial-time simulation and a non-interactive weakly CCA-secure commitment scheme.*

Proof. The first implication is immediate; since stand-alone SPS security is strictly weaker than externalized UC security, any NISC protocol satisfying externalized UC security is already stand-alone secure with SPS.

So, it suffices to prove that externalized UC-secure NISC implies weakly CCA-secure commitments; formally, we prove the following:

Input: The receiver R (with identity P_R) and the sender S (with identity P_S) are given input $x, y \in \{0,1\}^n$, respectively, and both parties have common input 1^n and session ID id.
Output: R outputs $f(x,y)$.

Round 1: R proceeds as follows:

1. Generate trapdoor $t \leftarrow \{0,1\}^n$ and randomness $r_x \leftarrow \{0,1\}^*$.
2. Compute $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$.
3. Compute $(\mathsf{msg}_1, \sigma_{\mathsf{NISC}}) \leftarrow \mathsf{NISC}_1(1^n, (x, r_x, t))$, where the protocol $\pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ computes the functionality h given in Figure 3.
4. Compute $(\mathsf{zk}_1, \sigma_{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_1(1^n)$.
5. Send $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$ to S.

Round 2: S proceeds as follows:

1. Generate randomness $r_1, r_2, r_{\mathsf{NISC}} \leftarrow \{0,1\}^*$.
2. Compute $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y, \perp, \perp); r_{\mathsf{NISC}})$.
3. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$, $w_1 = (r_{\mathsf{NISC}}, y)$, and $w_2 = (\perp, \perp, \perp)$. Compute $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_1; r_1)$ and $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_2)$.
4. Compute $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, \perp))$ for the language L consisting of tuples (v, c_1, c_2), where $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$, such that there exists a witness (w_1, r_1, w_2, r_2) so that either:
   (a) $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_1; r_1)$, and $w_1 = (r_{\mathsf{NISC}}, y)$ satisfies $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y, \perp, \perp); r_{\mathsf{NISC}})$.
   OR:
   (b) $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$, and $w_2 = (r_{\mathsf{NISC}}, t, z^*)$ satisfies $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \perp, t, z^*); r_{\mathsf{NISC}})$.
5. Send $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ to R.

Output phase: R proceeds as follows:

1. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$. If $\mathsf{ZK}_3(\mathsf{zk}_2, (v, c_1, c_2), \sigma_{\mathsf{ZK}}) \neq \mathsf{Accept}$, terminate with output $\perp$.
2. Compute $z = \mathsf{NISC}_3(\mathsf{msg}_2, \sigma_{\mathsf{NISC}})$. If $z = \perp$, terminate with output $\perp$; otherwise return z.

Fig. 2: Protocol Π for non-interactive secure computation.

Input: The receiver R has input (x, r_x, t), and the sender S has input (c_x, y, t', z^*)
Output: Either $f(x, y)$, z^*, or the special symbol $\perp$.

Functionality:

1. If $c_x \neq \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$, return $\perp$.
2. If $t = t'$, then return z^*.
3. Otherwise, return $f(x, y)$ (or $\perp$ if either x or y is $\perp$).

Fig. 3: Functionality h used for the underlying 2PC protocol π.

Lemma 1. *Assume a protocol $\Pi = (\pi_1, \pi_2, \pi_3)$ for non-interactive computation of the functionality which, on inputs x and y, returns $f(x, y) = 1$ if $x = y$ and $f(x, y) = 0$ otherwise; further assume that Π satisfies the notion of UC security with a superpolynomial-time helper. Then there exists a commitment scheme $(\mathsf{Com}, \mathsf{Open})$ which satisfies correctness, perfect binding for sufficiently large inputs, and weak CCA security.*

Proof. We define the weakly CCA secure commitment scheme $(\mathsf{Com}, \mathsf{Open})$ as follows:

- $\mathsf{Com}(1^n, \mathsf{id}, x)$ generates random padding $p \leftarrow \{0,1\}^n$ and outputs $c \leftarrow \pi_1(1^n, (\mathsf{id}, 1), x||p)$ as well as the session identifier id.
  That is, c is the first (receiver's) message of a new instance of Π with receiver input x, padded by the random p, and session identifier id.
  Note: We shall assume throughout that the player identifiers in any instance of Π are equal to 1 for the sender and 2 for the receiver.
- $\mathsf{Open}(c, x, (p, r))$ outputs Reject if $c \neq \pi_1(1^n, (\mathsf{id}, 1), x||p; r)$, and otherwise recovers the receiver's state σ after π_1 and outputs $b \leftarrow \pi_3(\pi_2(c, x), \sigma)$.
  That is, Open *first verifies that the commitment c is validly generated with respect to the value x and the receiver's randomness; if not, it returns* Reject. *Otherwise, it returns the result (*Accept *if 1,* Reject *if 0) of running the sender of Π given the initial message c and sender's input x to produce a message m, and finally running the receiver of Π given m as the sender's message.*

Correctness of $(\mathsf{Com}, \mathsf{Open})$ will follow directly from the correctness of Π. For the other two properties, we prove the following claims:

Claim 1 *For all sufficiently large input sizes $|x|$, $(\mathsf{Com}, \mathsf{Open})$ satisfies perfect binding.*

We defer some details to the full version, but provide a high-level summary of the argument here.

Essentially, perfect binding will follow from the correctness and security of Π. Fix the simulator $\mathcal{S}$ (and superpolynomial-time helper $\mathcal{H}$) given by the definition

of $\mathcal{H}$-EUC security for Π as a secure implementation of the equality functionality, and assume for the sake of contradiction that there exists an infinite sequence of tuples (c, x, x') such that, for each such pair, $x \neq x'$ but there exist (p, r) and (p', r') for which $\mathsf{Open}(c, x, (p, r)) = \mathsf{Accept}$ and $\mathsf{Open}(c, x', (p', r')) = \mathsf{Accept}$ both with non-zero probability.

Then consider an environment $\mathcal{Z}$ which, on input x^*, will do as follows:

- Start an instance of Π with a corrupted receiver, session identifier id (and player identifiers 1 for the receiver and 2 for the sender), and input $x||p$ for the receiver and x^* for the sender.
- Substitute c for the receiver's first message to the honest sender, and receive the sender's response m.
- Run the standard final round π_3 of the receiver's protocol using m as the sender's message and r as the randomness to produce an output $\pi_3(m)|_r$.

If $x^* = x||p$, the output must be 1 by perfect correctness of Π in the real execution of this environment, so the same must hold with overwhelming probability in the ideal execution using $\mathcal{T}_f$. This in turn indicates that the simulator $\mathcal{S}$, when given c as the receiver's first message, extracts the output $x||p$ to send to the ideal functionality with overwhelming probability, as the ideal functionality must return 1 when comparing that output to the sender's input $x||p$.

However, if we consider a similar experiment to the above but using $x'||p'$ as the receiver's input rather than $x||p$ (and r' as the respective randomness), we can use the same logic to arrive at the conclusion that the input extracted by the simulator $\mathcal{S}$ from c and sent to the ideal functionality on behalf of the corrupted receiver is $x'||p'$ with overwhelming probability. Clearly, for sufficiently large $n \in \mathbb{N}$ (i.e., sufficiently large inputs x, x' in our infinite sequence of tuples (c, x, x')), this cannot be true simultaneously with the above fact; thus, by contradiction, $(\mathsf{Com}, \mathsf{Open})$ must satisfy perfect binding.

Claim 2 $(\mathsf{Com}, \mathsf{Open})$ *satisfies weak CCA security.*

Proof. Fix the simulator $\mathcal{S}$ and superpolynomial-time helper $\mathcal{H}$ implied by the definition of $\mathcal{H}$-EUC security of the protocol Π. Assume for the sake of contradiction that there exists an adversary $\mathcal{A}$ which can contradict the definition of weak CCA security (Definition 9). We first show that $\mathcal{A}$, which is by definition polynomial-time with oracle access to a weak CCA decommitment oracle $\mathcal{O}^*$, can also be effectively implemented in polynomial time with oracle access to the helper functionality $\mathcal{H}$.

Subclaim 1 *Any polynomial-time adversary $\mathcal{A}$ against weak CCA security with oracle access to the oracle $\mathcal{O}^*$ defined in Definition 9 can also be implemented in polynomial time using oracle access to the helper functionality $\mathcal{H}$ instead, with error at most negligible in the security parameter n of Π*[9]*, and with the*

[9] We comment that, while the implementation of $\mathcal{O}^*$ does not decommit successfully with probability 1, decommitting with overwhelming probability is sufficient as it creates at most a negligible error in the adversary's output in the CCA security game.

additional property that $\mathcal{H}$ will never be queried using a session identifier sid *that is the same as the identifier used in $\mathcal{A}$'s challenge commitment.*

Proof. Consider replacing each of $\mathcal{A}$'s queries to $\mathcal{O}^*$ by the following process, which runs in polynomial time given oracle access to $\mathcal{H}$:

- Receive a commitment c to decommit, with tag id.
- Start a new instance of Π with a corrupted receiver and session identifier id (and player identifiers 1 for the receiver and 2 for the sender).
- Run the simulator $\mathcal{S}$ (which uses the helper $\mathcal{H}$) on the respective instance of Π, substituting c for the corrupted receiver's message. $\mathcal{S}$ will generate an input $x^*||p$ to send to the ideal functionality; return x^* to $\mathcal{A}$.

We claim that, if the above process does not generate correct responses to all oracle queries with overwhelming probability (i.e., $1 - \nu(n)$ for some negligible $\nu(\cdot)$), then there exists an environment $\mathcal{Z}$ able to distinguish between the real and simulated executions with non-negligible probability.

First, we consider a number of "hybrid" oracles $\mathcal{O}_0, \mathcal{O}_1, \ldots$, where in $\mathcal{O}_i$ the first i queries are answered by the true oracle $\mathcal{O}^*$ and all other queries are answered by the procedure above. Assume then for the sake of contradiction that there exists some fixed randomness r for the CCA security adversary such that, in the respective instance of the security game, the poly-time implementation of $\mathcal{O}^*$ gives at least one incorrect decommitment with some non-negligible probability $1/p(n)$. Then there necessarily exists some $i \in \mathbb{N}$ such that the oracle's outputs in $\mathcal{O}_i$ and $\mathcal{O}_{i-1}$ differ with non-negligible probability $1/q(n)$ (since the adversary in the CCA security game is restricted to at most a polynomial number of oracle queries).

We use this fact to construct our distinguishing environment $\mathcal{Z}$. Specifically, because of the above, there must exist $j \geq i$ for which the oracle's responses to the j^{th} query differ between $\mathcal{O}_i$ and $\mathcal{O}_{i-1}$ with some non-negligible probability $1/q'(n)$; let $\mathcal{Z}$ receive as non-uniform advice the first such j, the j^{th} query c, and the (padded) decommitment $x||p$ (which can be $\perp$ if c is an invalid commitment), which are determined by fixed randomness r and the responses from the true CCA oracle to the first $j-1$ queries, and let it proceed as follows:

- Start a single instance of Π with a corrupted receiver, session identifier given by the tag of c (and player identifiers 1 for the receiver and 2 for the sender), and receiver and sender input both equal to $x||p$.
- Replace the receiver's first message with c, and return the output of the protocol.

By perfect correctness of Π, and the assumption that c is a valid first-round message on input $x||p$, $\mathcal{Z}$ outputs 1 in the real interaction with probability 1; however, by our assumption that the responses to the j^{th} oracle query in $\mathcal{O}_i$ and $\mathcal{O}_{i-1}$ differ with non-negligible probability $1/q'(n)$, we know that in the ideal interaction $\mathcal{S}$ must send some $x'||p' \neq x||p$ to the ideal functionality on behalf of the corrupted receiver with at least probability $1/q'(n)$. Therefore, since the

honest sender's input to the ideal functionality is always $x||p$, we observe that $\mathcal{Z}$ outputs 0 in the ideal interaction with probability $1/q'(n)$, thus contradicting security of Π by distinguishing the real and ideal interactions and completing our argument.

Lastly, we note that, during the $\mathcal{H}$-aided reimplementation of the adversary $\mathcal{A}$, $\mathcal{H}$ will never be queried using a session identifier sid that is the same as the identifier used in the challenge commitment. This follows from the restriction that the simulator $\mathcal{S}$ may never query $\mathcal{H}$ using an honest party's identifiers $(\mathsf{sid}, \mathsf{pid})$: the only corrupted parties are those with sid equal to the tags of the queried commitments, which by the definition of weak CCA security may never be identical to the tag of the challenge. □

In the full version, we also show the following, which together with the previous claim will provide a contradiction:

Subclaim 2 $(\mathsf{Com}, \mathsf{Open})$ *satisfies hiding against any polynomial-time adversary $\mathcal{A}$, even if the adversary is given oracle access to the helper functionality $\mathcal{H}$, as long as $\mathcal{A}$ never queries $\mathcal{H}$ using a session identifier* sid *that is the same as the identifier used in the challenge commitment.*

So, given an adversary $\mathcal{A}$ that contradicts weak CCA security using polynomial time and oracle access to the CCA oracle $\mathcal{O}^*$, Subclaim 1 implies that there is a reimplemented adversary $\mathcal{A}'$ that likewise contradicts weak CCA security and uses polynomial time and oracle access to the superpolynomial-time helper functionality $\mathcal{H}$ without invoking the helper using a session identifier equal to the tag of the challenge commitment. But this directly contradicts Subclaim 2, since weak CCA security without access to the CCA oracle is equivalent to hiding, and the subclaim shows that $\mathcal{A}'$ cannot break the hiding property of $(\mathsf{Com}, \mathsf{Open})$ without invoking $\mathcal{H}$ using the challenge commitment's tag. Therefore, by this contradiction, $(\mathsf{Com}, \mathsf{Open})$ satisfies weak CCA security, as desired.

□
□
□

References

1. Abdolmaleki, B., Malavolta, G., Rahimi, A.: Two-Round Concurrently Secure Two-Party Computation. Cryptology ePrint Archive, Paper 2021/1357 (2021), https://eprint.iacr.org/2021/1357
2. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (Aug 2015). https://doi.org/10.1007/978-3-662-48000-7_32
3. Badrinarayanan, S., Garg, S., Ishai, Y., Sahai, A., Wadia, A.: Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 275–303. Springer, Heidelberg (Dec 2017). https://doi.org/10.1007/978-3-319-70700-6_10

4. Badrinarayanan, S., Goyal, V., Jain, A., Khurana, D., Sahai, A.: Round optimal concurrent MPC via strong simulation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 743–775. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_25
5. Barak, B., Sahai, A.: How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In: 46th FOCS. pp. 543–552. IEEE Computer Society Press (Oct 2005). https://doi.org/10.1109/SFCS.2005.43
6. Beaver, D.: Foundations of secure interactive computing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 377–391. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_31
7. Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 500–532. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_17
8. Benhamouda, F., Lin, H., Polychroniadou, A., Venkitasubramaniam, M.: Two-round adaptively secure multiparty computation from standard assumptions. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 175–205. Springer, Heidelberg (Nov 2018). https://doi.org/10.1007/978-3-030-03807-6_7
9. Bitansky, N., Lin, H.: One-message zero knowledge and non-malleable commitments. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 209–234. Springer, Heidelberg (Nov 2018). https://doi.org/10.1007/978-3-030-03807-6_8
10. Brakerski, Z., Döttling, N.: Two-message statistically sender-private OT from LWE. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 370–390. Springer, Heidelberg (Nov 2018). https://doi.org/10.1007/978-3-030-03810-6_14
11. Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. J. Comput. Syst. Sci. **37**(2), 156–189 (Oct 1988)
12. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology **13**(1), 143–202 (Jan 2000). https://doi.org/10.1007/s001459910006
13. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001). https://doi.org/10.1109/SFCS.2001.959888
14. Canetti, R., Dodis, Y., Pass, R., Walfish, S.: Universally composable security with global setup. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 61–85. Springer, Heidelberg (Feb 2007). https://doi.org/10.1007/978-3-540-70936-7_4
15. Canetti, R., Kushilevitz, E., Lindell, Y.: On the limitations of universally composable two-party computation without set-up assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 68–86. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_5
16. Canetti, R., Lin, H., Pass, R.: Adaptive hardness and composable security in the plain model from standard assumptions. In: 51st FOCS. pp. 541–550. IEEE Computer Society Press (Oct 2010). https://doi.org/10.1109/FOCS.2010.86
17. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (1998), manuscript
18. Döttling, N., Garg, S., Hajiabadi, M., Masny, D., Wichs, D.: Two-round oblivious transfer from CDH or LPN. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 768–797. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_26

19. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: 30th ACM STOC. pp. 409–418. ACM Press (May 1998). https://doi.org/10.1145/276698.276853
20. Feige, U.: Alternative Models for Zero-Knowledge Interactive Proofs. Ph.D. Thesis, Weizmann Institute of Science (1990)
21. Fernando, R., Jain, A., Komargodski, I.: Maliciously-Secure MrNISC in the Plain Model. Cryptology ePrint Archive, Paper 2021/1319 (2021), https://eprint.iacr.org/2021/1319
22. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 467–476. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488667
23. Garg, S., Goyal, V., Jain, A., Sahai, A.: Concurrently secure computation in constant rounds. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 99–116. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_8
24. Garg, S., Kiyoshima, S., Pandey, O.: On the exact round complexity of self-composable two-party computation. In: Coron, J., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 194–224. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56614-6_7
25. Garg, S., Srinivasan, A.: Two-round multiparty secure computation from minimal assumptions. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 468–499. Springer, Heidelberg (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_16
26. Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge, UK (2001)
27. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987). https://doi.org/10.1145/28395.28420
28. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology **7**(1), 1–32 (Dec 1994). https://doi.org/10.1007/BF00195207
29. Goldwasser, S., Levin, L.A.: Fair computation of general functions in presence of immoral majority. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO'90. LNCS, vol. 537, pp. 77–93. Springer, Heidelberg (Aug 1991). https://doi.org/10.1007/3-540-38424-3_6
30. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
31. Goyal, V., Lin, H., Pandey, O., Pass, R., Sahai, A.: Round-efficient concurrently composable secure computation via a robust extraction lemma. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 260–289. Springer, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46494-6_12
32. Kiyoshima, S.: Round-efficient black-box construction of composable multi-party computation. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part II. LNCS, vol. 8617, pp. 351–368. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44381-1_20
33. Kiyoshima, S., Manabe, Y., Okamoto, T.: Constant-round black-box construction of composable multi-party computation protocol. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 343–367. Springer, Heidelberg (Feb 2014). https://doi.org/10.1007/978-3-642-54242-8_15

34. Lin, H., Pass, R., Seth, K., Telang, S.: Output-compressing randomized encodings and applications. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part I. LNCS, vol. 9562, pp. 96–124. Springer, Heidelberg (Jan 2016). https://doi.org/10.1007/978-3-662-49096-9_5
35. Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In: Umans, C. (ed.) 58th FOCS. pp. 576–587. IEEE Computer Society Press (Oct 2017). https://doi.org/10.1109/FOCS.2017.59
36. Lin, H., Pass, R., Venkitasubramaniam, M.: Concurrent non-malleable commitments from any one-way function. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 571–588. Springer, Heidelberg (Mar 2008). https://doi.org/10.1007/978-3-540-78524-8_31
37. Lin, H., Pass, R., Venkitasubramaniam, M.: A unified framework for concurrent security: universal composability from stand-alone non-malleability. In: Mitzenmacher, M. (ed.) 41st ACM STOC. pp. 179–188. ACM Press (May / Jun 2009). https://doi.org/10.1145/1536414.1536441
38. Lindell, Y.: Lower bounds for concurrent self composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_12
39. Malkin, T., Moriarty, R., Yakovenko, N.: Generalized environmental security from number theoretic assumptions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 343–359. Springer, Heidelberg (Mar 2006). https://doi.org/10.1007/11681878_18
40. Micali, S., Rogaway, P.: Secure computation (abstract). In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 392–404. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_32
41. Morgan, A., Pass, R., Polychroniadou, A.: Succinct non-interactive secure computation. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 216–245. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45724-2_8
42. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 57–74. Springer, Heidelberg (Aug 2008). https://doi.org/10.1007/978-3-540-85174-5_4
43. Pass, R.: Simulation in quasi-polynomial time, and its application to protocol composition. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 160–176. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_10
44. Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: 46th FOCS. pp. 563–572. IEEE Computer Society Press (Oct 2005). https://doi.org/10.1109/SFCS.2005.27
45. Prabhakaran, M., Sahai, A.: New notions of security: Achieving universal composability without trusted setup. In: Babai, L. (ed.) 36th ACM STOC. pp. 242–251. ACM Press (Jun 2004). https://doi.org/10.1145/1007352.1007394
46. Rackoff, C., Simon, D.R.: Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 433–444. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_35
47. Schröder, D., Unruh, D.: Round optimal blind signatures. Cryptology ePrint Archive, Report 2011/264 (2011), https://eprint.iacr.org/2011/264
48. Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd FOCS. pp. 160–164. IEEE Computer Society Press (Nov 1982). https://doi.org/10.1109/SFCS.1982.38