Concurrently Composable Non-Interactive Secure Computation*

Andrew Morgan
Cornell University
asmorgan@cs.cornell.edu

Rafael Pass†
Cornell Tech
rafael@cs.cornell.edu

November 28, 2021

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.

*The original version of this paper which only included the feasibility results was first submitted to a conference on May 21, 2020. The necessity of the assumptions was first claimed with a proof sketch in a conference submission on May 27, 2021. The current version includes the full proofs of the necessary conditions.

†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.

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 the output $f(x, y)$ of the function. Such 2-round protocols are referred to as *non-interactive secure computation protocols (NISC)*.[1]

Security of secure computation protocols is traditionally defined using the *simulation paradigm*, first introduced in [28] and extended in several later works [27, 5, 38, 11]. 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., [38, 11]) 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 [19, 16, 18]) 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 [38, 11] 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 [12]. 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 interactions with π from simulated interactions with the idealized g, we can effectively replace π with the idealized g when proving Π secure.

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

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 [14]—or, disregarding composability, even concurrently securely [36]—without introducing additional *trusted setup* assumptions.

The notion of *superpolynomial-time simulation* (SPS) [41], 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 [41, 45, 2, 39] and the more general case of *multi-party computation* [4, 35, 21], 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* [43] and its generalization of *"UC-security with a superpolynomial-time helper"* [15] 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 [43, 37, 15, 30, 31, 29]; the most recent constructions have been based on the notion of "CCA-secure" commitments [15], 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 [26]. 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 [41, 2, 39]. (Indeed, enabling secure 2-round protocols was one of the original motivations behind the notion of SPS security [41].) 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:

- [21] proposed the first concurrently secure constant-round protocol based on standard assumptions, and this bound was later reduced to 5 [22].
- [3] 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., [6, 23, 7]) require a common reference string (CRS) as "trusted setup".
- For the special case of zero-knowledge arguments of knowledge, [41] 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* [16].)

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

Meanwhile, for composable "angel-based" security in the plain model, the situation is even worse; the protocol proposed by [15] requires n^ϵ rounds, while [30] reduced this to logarithmic round complexity and [31] 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*![2]

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* [40, 15, 33, 8].
- A *stand-alone secure SPS-NISC* with subexponential security [2].

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 [44]). Our main result, then, is as follows:

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

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

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[3]:

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 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 [2] or subexponential hardness of the Learning With Errors assumption [9].

Non-interactive CCA secure commitments, however, require more complex assumptions. They were first constructed in [40] based on adaptive one-way permutations; later, [33] 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-indistinguishable arguments (NIWIs), and subexponentially-secure time-lock puzzles. Even more recently, [8] 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" [42, 34] (and not the stronger notion of CCA security), it seems that a relatively minor modification of their analysis (similar to the analysis in [33]) 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, and prove it in Section 5. In addition, we formalize

[3] 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.

and prove Theorem 2 in Section 6.

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* [11, 13], 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 [43] for the case of non-interactive, stateless, angels) In our case (as in [15]) $\mathcal{H}$ 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 sub-components. 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 [39], by using the underlying NISC to implement an "interactive witness encryption scheme".[4] 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 $\mathcal{H}$.

Dealing with a malicious sender: using a "two-track" functionality For the case where the sender is corrupted, we begin by having the sender produce a commitment c_y to their input y and send it along with their second message. Unlike with c_x, however, we cannot verify c_y within the underlying NISC protocol, since the receiver does not receive the commitment c_y until after they have given their inputs to the NISC (and hence cannot provide it as an input to verify that the commitment they received from the sender is correct). Instead, we rely on the SPS-ZK argument, which the sender uses to prove that c_y is a valid commitment to the same input y used to produce their NISC message. However, this in turn creates issues for simulatability in the corrupted-receiver case, since the simulator does not know y and cannot simulate the underlying NISC or the ZK argument in only polynomial time.

To deal with this, we switch to what is effectively a *two-track* functionality for the underlying NISC and ZK argument. We add a trapdoor t, chosen at random and committed to by the receiver simultaneously with x. Furthermore, 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 c_y and the sender's NISC message are correctly generated (i.e., using the same 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). In fact, this is not quite sufficient to simulate for a corrupted sender; 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 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

[4] Recall that *witness encryption* [20] 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".

extractability by using a technique from [41] 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, we note at this point that the commitment c_y is actually redundant, since the sender already commits to y as part of their commitment to the witness w_1, and, as we observed, the corrupted sender must (unless they can correctly guess the trapdoor t) use the witness w_1 to produce an accepting SPS-ZK proof. Hence, we can remove c_y entirely and have the simulator extract the corrupted sender's input from c_1 using the helper $\mathcal{H}$ instead.

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. [1, 32]. 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 [17].

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 the 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.[5]

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:

[5] 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.

- 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).

Definition 1 ([39], based on [46, 25, 2])**.** 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.
- $\mathit{out} \leftarrow \mathsf{NISC}_3(\sigma, \mathsf{msg}_2)$ 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:

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

Externalized Universally Composable Security. To define the notion of security proven in our main theorem, we use the framework of *universally composable security* [11, 12], extended to include access to superpolynomial "helper functionalities" [13, 15]. 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 [15]. Consider parties represented by polynomial-time interactive Turing machines [28]; 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" [43], which can be formalized in the *externalized UC* (EUC) setting [13]; 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 [15])**.** 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 $z \in \{0,1\}^*$:

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

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 [15])**.** 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. [26] 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 [41, 43]). 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 [11, 15])**.** 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 [39]) 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 ([39], based on [47, 25, 41, 43, 2])**.** Given a function $T(\cdot)$, a non-inter-active 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:

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

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. [2] 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 [2, 9]:

Theorem 4 ([2, 9])**.** 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 ([10, 28, 24])**.** 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 $x \in \mathcal{L}$:

$$\Pr\left[\langle P, V\rangle(x) = \mathsf{Accept}\right] \geq 1 - \nu(|x|)$$

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 $x \notin \mathcal{L}$:

$$\Pr\left[\langle P^*, V\rangle(x) = \mathsf{Accept}\right] \leq \nu(|x|)$$

   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* [28] with superpolynomial simulation (SPS-ZK) [41], 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 ([41])**.** 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:

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

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 *stand-alone 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 stand-alone 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 Appendix A.

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 [40, 15, 33].

Definition 8 (based on [33])**.** 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 any commitment string c, values v, v', and randomness r, r', if it is true that $\mathsf{Open}(c, v, r) = 1$ and $\mathsf{Open}(c, v', r') = 1$, then $v = v'$.
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 [44]. 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 6, 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.

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 [44].

4 Results

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

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

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 $\perp$).

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

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 next section.

5 Proof

Correctness of Π follows trivially from the correctness of π and $(\mathsf{Com}, \mathsf{Open})$. To prove security, we state the following lemma, which amounts to showing that Π $\mathcal{H}$-EUC-realizes the ideal functionality $\mathcal{T}_f$ for secure two-party computation of $f(\cdot, \cdot)$:

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.

Figure 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 $\perp$).

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

Lemma 1. Let $\Pi(\mathcal{T}_f)$ be the protocol where the sender and the receiver send their inputs x and y to the ideal functionality $\mathcal{T}_f$, which, given those inputs, computes $f(x, y)$ and sends the result to the receiver.

Then, for any polynomial-time adversary $\mathcal{A}$, there exists a polynomial-time simulator $\mathcal{S}$ such that, for any non-uniform polynomial-time environment $\mathcal{Z}$ and polynomial-time distinguisher D, there exists a negligible function $\nu(\cdot)$ such that, for any $n \in \mathbb{N}$ and $\mathsf{aux} \in \{0,1\}^*$:

$$\left|\Pr\left[D(\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(1^n, \mathsf{aux})) = 1\right] - \Pr\left[D(\mathsf{Exec}_{\Pi(\mathcal{T}_f),\mathcal{S},\mathcal{Z}}(1^n, \mathsf{aux})) = 1\right]\right| \leq \nu(n)$$

Proof. Recall that the environment and adversary will have access to the helper $\mathcal{H}$ in both the real and ideal experiments, as will the simulator $\mathcal{S}$ which we now construct. $\mathcal{S}$ will forward communication directly between the environment $\mathcal{Z}$ and the helper $\mathcal{H}$, while simulating the session of Π started by $\mathcal{Z}$ by running one of the simulators $\mathcal{S}_R$ (see Figure 4), $\mathcal{S}_S$ (see Figure 5), or $\mathcal{S}_N$ (see Figure 6), depending on whether (respectively) the receiver, the sender, or neither is corrupted. We can then demonstrate that the environment's interaction with the simulator $\mathcal{S}$ in the ideal world is indistinguishable from the environment's interaction with the adversary $\mathcal{A}$ (which, without loss of generality, we can assume to be controlled by the environment) in the real world.

So, assume for the sake of contradiction that there exist some environment $\mathcal{Z}$ and distinguisher D such that, for infinitely many $n \in \mathbb{N}$, there is auxiliary input $\mathsf{aux} \in \{0,1\}^n$ such that D distinguishes the distributions $\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}}(1^n, \mathsf{aux})$ and $\mathsf{Exec}_{\Pi(\mathcal{T}_f),\mathcal{S},\mathcal{Z}}(1^n, \mathsf{aux})$ with some non-negligible probability $1/p(n)$ (for polynomial $p(\cdot)$). We henceforth denote the experiments which produce these distributions by $\mathsf{Exp}_{\mathsf{Real}}$ and $\mathsf{Exp}_{\mathsf{Sim}}$, respectively; it then suffices to show that their respective views $\mathsf{View}_{\mathsf{Real}}$ and $\mathsf{View}_{\mathsf{Sim}}$ cannot be distinguished by any distinguisher running in polynomial time with access to the helper $\mathcal{H}$. We handle this in three separate cases, depending on whether the receiver, sender, or neither party is corrupted in the session of Π.

5.1 Corrupted Receiver

In this case, the simulator $\mathcal{S}$ will run $\mathcal{S}_R$ as given in Figure 4 in the experiment $\mathsf{Exp}_{\mathsf{Sim}}$; we prove that there exists no distinguisher between the views $\mathsf{View}_{\mathsf{Real}}$ and $\mathsf{View}_{\mathsf{Sim}}$ by a sequence of hybrids:

- Let H_0 be identical to $\mathsf{Exp}_{\mathsf{Real}}$, with the exception that the honest sender sends y to an instance of the ideal functionality $\mathcal{T}_f$, uses the decommitment helper $\mathcal{H}$ to retrieve the values

Simulator $\mathcal{S}_R$

1. Receive the adversary's first-round message $(\mathsf{msg}_1, \mathsf{zk}_1, c_x)$.
2. Generate randomness $r_{\mathsf{NISC}} \leftarrow \{0,1\}^*$.
3. Use the helper $\mathcal{H}$ to compute $x^*||t \leftarrow \mathcal{H}(c_x)$.
4. Send x^* to the ideal functionality $\mathcal{T}_f$ and receive the output z.
5. Compute $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \bot, t, z); r_{\mathsf{NISC}})$.
6. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$, $w_1 = (\bot, \bot)$, and $w_2 = (r_{\mathsf{NISC}}, t, z)$. Compute $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_1)$ and $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$.
7. Compute $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, \bot, w_2, r_2))$.
8. Send $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ to the adversary.

Figure 4: Simulator for a corrupted receiver.

$x^*||t \leftarrow \mathcal{H}(c_x)$, sends x^* to $\mathcal{T}_f$ on behalf of the receiver, and retrieves the output z from $\mathcal{T}_f$.

Views are identically distributed.

- Let H_1 be identical to H_0, with the exception that the sender computes the commitment c_2 as $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$ (where $w_2 = (r_{\mathsf{NISC}}, t, z)$), rather than as $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_2)$.

  Follows from weak CCA security of $(\mathsf{Com}, \mathsf{Open})$.

- Let H_2 be identical to H_1, with the exception that the sender computes the second ZK message as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$ rather than computing it as $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, \bot))$.

  Follows from simulation-based security of the ZK proof, specifically indistinguishability against $T_{\mathsf{ZK}}(\cdot)$*-time distinguishers and* $T_{\mathsf{ZK}}(\cdot) \gg T_{\mathsf{Com}}(\cdot)$.

- Let H_3 be identical to H_2, with the exception that the sender computes the second NISC message using the corrupted-receiver simulator $\mathcal{S}'_R$ for the underlying NISC π, as follows:

  - Forward the corrupted receiver's first-round message msg_1 for π to $\mathcal{S}'_R$, which will produce a message (x, r_x, t) to send to the ideal functionality $\mathcal{T}_h$.
  - Verify that the corrupted receiver's commitments c_x and c_t satisfy $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$; if not, forward $\bot$ to $\mathcal{S}'_R$ as the result from $\mathcal{T}_h$.
  - Forward $f(x^*, y)$ to $\mathcal{S}'_R$ as the result from $\mathcal{T}_h$.
  - $\mathcal{S}'_R$ will produce a message msg'_2 to send to the corrupted receiver for π; send $(\mathsf{msg}'_2, \mathsf{zk}_2, c_1, c_2)$ to the corrupted receiver for Π.

  rather than computing it as $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y, \bot, \bot))$.

  Follows from simulation-based security of the NISC protocol π*, specifically indistinguishability against* $T_\pi(\cdot)$*-time distinguishers and* $T_\pi(\cdot) \gg T_{\mathsf{Sim}}(\cdot) + T_{\mathsf{Com}}(\cdot)$.

- Let H_4 be identical to H_3, with the exception that the sender computes the second NISC message as $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \bot, t, z))$—that is, using the trapdoor—rather than computing it using the simulator $\mathcal{S}'_R$ as in H_3.

  Follows again from simulation-based security of the NISC protocol π*, and additionally the fact that the simulator* $\mathcal{S}'_R$ *depends only on the adversary and not the inputs to the NISC (hence, the same simulator can be used for the definition of security in this hybrid as in the last).*

- Let H_5 be identical to H_4, with the exception that the sender computes the second ZK message as $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{zk}_1, (v, c_1, c_2), (w_1, \perp, w_2, r_2))$, where $w_1 = (\perp, \perp)$ and $w_2 = (r_{\mathsf{NISC}}, t, z)$, rather than computing it as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$.

  Follows once again from simulation-based security of the ZK proof.

- Let H_6 be identical to H_5, with the exception that the sender computes the commitment c_1 as $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_1)$, rather than as $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_1; r_1)$ (where $w_1 = (r_{\mathsf{NISC}}, y)$). Note that H_6 is now identical to the experiment $\mathsf{Exp}_{\mathsf{Sim}}$ where the adversary interacts with the simulator $\mathcal{S}_R$.

  Follows from weak CCA security of $(\mathsf{Com}, \mathsf{Open})$.

We set out to prove:

Claim 1. For the case where the receiver is corrupted in the session of Π, there exists no polynomial-time distinguisher D^* with oracle access to $\mathcal{H}$ such that, for some polynomial $p^*(\cdot)$:

$$|\Pr[D^*(\mathsf{View}_{\mathsf{Real}}) = 1] - \Pr[D^*(\mathsf{View}_{\mathsf{Sim}}) = 1]| \geq 1/p^*(n)$$

Proof. Observe that, if the views $\mathsf{View}_{\mathsf{Real}}$ and $\mathsf{View}_{\mathsf{Sim}}$ are distinguishable (i.e., the claim is not true), there must be some sequence of randomness r for the experiments prior to the session of Π (recall that the experiments prior to Π are identical) such that the views $\mathsf{View}_{\mathsf{Real}}|_r$ and $\mathsf{View}_{\mathsf{Sim}}|_r$, henceforth denoting the respective views with randomness fixed to r when applicable, are similarly distinguishable—that is:

$$|\Pr[D^*(\mathsf{View}_{\mathsf{Real}}|_r) = 1] - \Pr[D^*(\mathsf{View}_{\mathsf{Sim}}|_r) = 1]| \geq 1/p^*(n)$$

We thus proceed by proving the respective hybrids are indistinguishable for any such fixed randomness r.

To start, the views of H_0 and $\mathsf{Exp}_{\mathsf{Real}}$ are trivially identically distributed as the sender computes all of its messages in the same way. We continue with the following claims to complete the proof:

Subclaim 1. There exists no polynomial-time distinguisher D with access to the decommitment helper $\mathcal{H}$ such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_0]|_r) = 1] - \Pr[D(\mathsf{View}[H_1]|_r) = 1]| \geq 1/p'(n)$$

Proof. This will follow from the CCA security of $(\mathsf{Com}, \mathsf{Open})$. Assuming for the sake of contradiction that there exists a polynomial-time distinguisher D which distinguishes $\mathsf{View}[H_0]|_r$ and $\mathsf{View}[H_1]|_r$ with non-negligible probability $1/p'(n)$, we can use this to construct a polynomial-time adversary $\mathcal{A}_{\mathsf{Com}}$ which breaks the CCA security of the commitment scheme. $\mathcal{A}_{\mathsf{Com}}$ does as follows:

- Run the experiment given by H_0 with the following differences:
  - Use the fixed randomness r prior to the session of Π.

- Use CCA oracle queries in place of queries to the decommitment helper $\mathcal{H}$ for commitments using corrupted parties' identifiers.
- Respond with $\perp$ to commitment queries using honest parties' identifiers.
- When the honest sender generates the witness $w_2 = (r_{\mathsf{NISC}}, t, z)$, send the values w_2 and 0 and the tag (id, P_S) to the challenger to receive a commitment c^* of a randomly chosen one of the two values.
- Substitute c^* for c_2 whenever it occurs in the session of Π.

- Once the experiment finishes, run the distinguisher D on the final view and output the result.

Since S is honest, $\mathcal{A}_{\mathsf{Com}}$ never makes a CCA oracle query using the challenge commitment's tag (id, P_S). Furthermore, we note that the ZK proof zk_2 is independent of the value of w_2 (as it is computed using $w_2 = (\perp, \perp, \perp)$). Hence, if c^* is a commitment to 0, the view of the experiment is identically distributed to $\mathsf{View}[H_0]|_r$; if it is instead a commitment to w_2, the view is identically distributed to $\mathsf{View}[H_1]|_r$. Hence, if D distinguishes between $\mathsf{View}[H_0]|_r$ and $\mathsf{View}[H_1]|_r$ with probability $1/p'(n)$, $\mathcal{A}_{\mathsf{Com}}$ can distinguish between the experiments Exp_0 and Exp_1 in the CCA security game with probability $1/p'(n)$ and without making queries to commitments with the challenge tag, contradicting the CCA security of $(\mathsf{Com}, \mathsf{Open})$. □

Subclaim 2. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_1]|_r) = 1] - \Pr[D(\mathsf{View}[H_2]|_r) = 1]| \geq 1/p'(n)$$

Proof. This will follow from the simulation-based security of the ZK proof $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$. Assume for the sake of contradiction that there exists some $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D which distinguishes the respective views with some non-negligible probability $1/p'(n)$; in that case, we can construct a $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D' which can distinguish the real ZK proof zk_2 from the output of the simulator $\mathsf{Sim}_{\mathsf{ZK}}$ given a certain statement $v \in L$.

There must exist some assignment of the randomness r' for both the corrupted receiver and honest sender prior to the point where the ZK proof zk_2 is generated such that D distinguishes the distributions $\mathsf{View}[H_1]|_{r||r'}$ and $\mathsf{View}[H_2]|_{r||r'}$ with probability $1/p'(n)$. Moreover, the randomness r' defines a unique statement $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$ and commitments c_1, c_2, where we know $(v, c_1, c_2) \in L$ because it was generated honestly by the sender, as well as specific witnesses $w_1 = (r_{\mathsf{NISC}}, y)$ and $w_2 = (\perp, \perp, \perp)$ and a specific first message zk_1. We can then construct the $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D' which distinguishes between the distributions $\mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, \perp))$ and $\mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$. Specifically, given a sample zk_2^* from one of the two distributions, D' does as follows:

- Run the experiment given by H_1 until the session of Π, using the fixed randomness r. Internally run the $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$ in place of the decommitment helper $\mathcal{H}$ throughout the experiment. (This will require time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$.)
- For the session of Π, let both parties use the fixed randomness given by r', and replace the ZK proof zk_2 by the given zk_2^*.

- Once Π finishes, run D on the resulting view and output the result.

If zk_2^* is an honestly generated proof $\mathsf{zk}_2^* \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, r_2))$, then the view given to D is identically distributed to $\mathsf{View}[H_1]|_{r||r'}$; meanwhile, if it is a simulated proof $\mathsf{zk}_2^* \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$, then the view will be identically distributed to $\mathsf{View}[H_2]|_{r||r'}$. Hence D' distinguishes the real and simulated proofs with the same probability $1/p'(n)$ as with which D distinguishes the respective views of the hybrids, contradicting the definition of simulation-based security of the ZK proof since it runs in time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n) \ll T_{\mathsf{ZK}}(n)$. □

Notably, the above claim (in addition to future claims which prove non-existence of a $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher) applies identically to any polynomial-time distinguisher with oracle access to $\mathcal{H}$, since any such distinguisher can trivially be made into a $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher by running $\mathcal{O}$ internally in place of $\mathcal{H}$.

Subclaim 3. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_2]|_r) = 1] - \Pr[D(\mathsf{View}[H_3]|_r) = 1]| \geq 1/p'(n)$$

Proof. To prove this, we leverage the stand-alone security of the internal NISC protocol π. First, assume the opposite for the sake of contradiction—that is, that there exists such a $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D able to distinguish the views with probability $1/p'(n)$. The definition of stand-alone security guarantees that, for every adversary $\mathcal{A}'$ against π, there exists a simulator $\mathcal{S}'_R$ whose interaction with a corrupted receiver in an idealized experiment with the ideal functionality $\mathcal{T}_h$ is indistinguishable from the real interaction where parties use π. Notably, the simulator $\mathcal{S}'_R$ depends *only* on the adversary and not on the inputs to the NISC, so the same simulator applies to the adversary (i.e., the corrupted receiver) in H_2 as to the adversary in H_3, and, critically, later to the adversary in H_4.

Now, assuming without loss of generality that the adversaries are deterministic and take their randomness as part of the auxiliary input aux, there must exist auxiliary input aux such that D distinguishes $\mathsf{View}[H_2]|_{r||\mathsf{aux}}$ from $\mathsf{View}[H_3]|_{r||\mathsf{aux}}$ with probability $1/p'(n)$. In particular, we note that aux, in conjunction with the inputs x and y, fixes the corrupted receiver's first NISC message msg_1, as well as both inputs (x, r_x, t) and $(c_x, y, \perp, \perp)$ to π.

We construct a D' that runs in time $T_{\mathsf{Sim}}(n) \cdot \mathsf{poly}(n)$ and distinguishes the real and simulated executions of π (with inputs x and y and auxiliary input aux) on the respective inputs (x, r_x, t) and $(c_x, y, \perp, \perp)$. D', given some view (m_1, m_2, out) of the messages and receiver's output from either the real interaction (between the corrupted receiver and honest sender) or the ideal interaction (between the corrupted receiver and simulator $\mathcal{S}'_R$), does as follows:

- Run the experiment given by H_2 until the session of Π, using the fixed randomness r. Internally run the $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$ in place of the decommitment helper $\mathcal{H}$ throughout the experiment. (Since the experiment runs the ZK simulator $\mathsf{Sim}_{\mathsf{ZK}}$, this will require time $T_{\mathsf{Sim}}(n) + T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$.)
- For the execution of Π, let the adversary use the fixed randomness given by aux (note that this will fix their first NISC message msg_1 to be identical to m_1), and replace the honest sender's second NISC message msg_2 by m_2.

- Once Π finishes, run D on the resulting view and output the result.

If the given view is the real execution of the NISC protocol π, the view given to D will be identically distributed to $\mathsf{View}[H_2]|_{r||\mathsf{aux}}$; if the view is the simulated execution using $\mathcal{S}'_R$, then the view given to D will be identically distributed to $\mathsf{View}[H_3]|_{r||\mathsf{aux}}$. Hence, D' must distinguish the views of the real and simulated interactions with probability $1/p'(n)$.

We remark that it is critical that the output of h run on the respective inputs be the same between H_2 and H_3 so that the output of the (real or simulated) NISC in the views distinguished by D matches the respective output in the views distinguished by D'. Indeed, it can easily be verified that the output of h in both experiments is always $f(x, y)$ when the commitments c_x and c_t are validly generated and always $\perp$ if not.

Hence, the existence of a distinguisher D which distinguishes $\mathsf{View}[H_0]|_r$ and $\mathsf{View}[H_1]|_r$ with non-negligible probability implies a D' that contradicts the stand-alone security of π (since D' runs in time $T_{\mathsf{Sim}}(n) \cdot \mathsf{poly}(n) \ll T_\pi(n)$), completing the argument. □

Subclaim 4. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_3]|_r) = 1] - \Pr[D(\mathsf{View}[H_4]|_r) = 1]| \geq 1/p'(n)$$

Proof. Identical to Subclaim 3, except that D' now differentiates between the real and simulated executions of π on inputs (x, r_x, t) and $(c_x, \perp, t, z)$. We note that, as with the above experiment, the output of h is always identical between H_3 and H_4; furthermore, since the adversary $\mathcal{A}'$ remains the same throughout all of our hybrids, the same simulator $\mathcal{S}'_R$ satisfies the definition of security for the proof here as it does for the proof in the previous subclaim. □

Subclaim 5. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_4]|_r) = 1] - \Pr[D(\mathsf{View}[H_5]|_r) = 1]| \geq 1/p'(n)$$

Proof. Identical to Subclaim 2, but using different witnesses $w_1 = (\perp, \perp)$ and $w_2 = (r_{\mathsf{NISC}}, t, z)$, though, importantly, the statement (v, c_1, c_2) is the same. □

Subclaim 6. There exists no polynomial-time distinguisher D with access to the decommitment helper $\mathcal{H}$ such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_5]|_r) = 1] - \Pr[D(\mathsf{View}[H_6]|_r) = 1]| \geq 1/p'(n)$$

Proof. Identical to Subclaim 1, but using the values w_1 and 0 for the commitment c_1. Note that the ZK proof zk_2 is independent of the value of w_1 (as it is computed using $w_1 = (\perp, \perp)$). □

Simulator $\mathcal{S}_S$

1. Generate trapdoor $t \leftarrow \{0,1\}^n$ and randomness $r_x \leftarrow \{0,1\}^*$.
2. Compute $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), 0||t; r_x)$.
3. Compute $(\mathsf{msg}_1, \sigma_{\mathsf{NISC}}) \leftarrow \mathsf{NISC}_1(1^n, (0, r_x, t))$.
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 the adversary.
6. Receive output $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ from the adversary.
7. 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 $\perp$.
8. Use the helper to decommit $w_1 \leftarrow \mathcal{H}(c_1)$. Parse $w_1 = (r_{\mathsf{NISC}}, y^*)$.
9. Verify that $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y^*, \perp, \perp); r_{\mathsf{NISC}})$; if not, terminate with output $\perp$.
10. Send y^* to the ideal functionality $\mathcal{T}_f$.

Figure 5: Simulator for a corrupted sender.

Since the view of H_6 is by inspection identically distributed to the simulated experiment $\mathsf{Exp}_{\mathsf{Sim}}$, this suffices to complete the proof of the overall claim by a standard hybrid argument (i.e., if there were a D^* contradicting the claim, it would necessarily also contradict one of the above subclaims 1-6 by distinguishing the respective distributions with probability $1/6p^*(n)$).

□

5.2 Corrupted Sender

In this case, the simulator $\mathcal{S}$ will run $\mathcal{S}_S$ as given in Figure 5 in the experiment $\mathsf{Exp}_{\mathsf{Sim}}$; we again use a sequence of hybrids:

- Let H_0 be identical to $\mathsf{Exp}_{\mathsf{Real}}$, with the exception that the honest receiver sends x to an instance of the ideal functionality $\mathcal{T}_f$, uses the decommitment helper $\mathcal{H}$ to retrieve the value $w_1 \leftarrow \mathcal{H}(c_1)$, parses $w_1 = (r_{\mathsf{NISC}}, y^*)$ and sends y^* to $\mathcal{T}_f$ on behalf of the sender (or $\perp$ if w_1 fails to parse).

  Views are identically distributed.

- Let H_1 be identical to H_0, with the exception that the receiver, rather than using NISC_1 and NISC_3 to honestly generate the NISC messages and output, instead uses the corrupted sender simulator $\mathcal{S}'_S$ for the underlying NISC π and the corrupted sender $\mathcal{A}$. Specifically, in H_1, the receiver:

  - Sends the simulated message msg'_1 from $\mathcal{S}'_S$ to the corrupted sender instead of msg_1.
  - Upon receiving the corrupted sender's message, verifies that:
    * $w_1 \leftarrow \mathcal{H}(c_1)$ parses as $w_1 = (r_{\mathsf{NISC}}, y^*)$,
    * $\mathsf{ZK}_3(\mathsf{zk}_2, (v, c_1, c_2), \sigma_{\mathsf{ZK}}) = \mathsf{Accept}$, and
    * $\mathsf{msg}_2 = \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, y^*, \perp, \perp); r_{\mathsf{NISC}})$.

    Terminates with output $\perp$ if not true.
  - Otherwise, returns the output z from $\mathcal{T}_f$.

  The real and simulated first messages are indistinguishable by the simulation-based security of π with respect to $T_\pi(\cdot)$-time distinguishers. Furthermore, by soundness of the ZK proof (with respect to $T_{\mathsf{ZK}}(\cdot)$-time adversaries) and weak CCA security of the commitment scheme,

the corrupted sender, if it provides an accepting proof, must with overwhelming probability provide a valid witness to part (a) of the ZK language; in that case, we can show that the outputs are identical.

- Let H_2 be identical to H_1, with the exception that the receiver computes the commitment to x as $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), 0||t; r_x)$ rather than $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$.

  Follows from weak CCA security of the commitment scheme $(\mathsf{Com}, \mathsf{Open})$*; however, the CCA security adversary cannot run the full simulator* $\mathcal{S}'_S$*, so instead the simulated first message of the NISC is hard-coded as non-uniform advice. Note that, since the first message of the NISC is simulated, the receiver's first message is no longer dependent on* x*.*

- Let H_3 be identical to H_2, with the exception that the receiver no longer uses the simulator $\mathcal{S}'_S$ and instead computes $(\mathsf{msg}_1, \sigma_{\mathsf{NISC}}) \leftarrow \mathsf{NISC}_1(1^n, (0, r_x, t))$. Note that H_3 is now identical to the experiment $\mathsf{Exp}_{\mathsf{Sim}}$ where the adversary interacts with the simulator $\mathcal{S}_S$ and the honest receiver outputs the result z from the ideal functionality $\mathcal{T}_f$.

  Follows from the simulation-based security of π*.*

We claim the following:

Claim 2. For the case where the sender is corrupted in the session of Π, there exists no polynomial-time distinguisher D^* with oracle access to $\mathcal{H}$ such that, for some polynomial $p^*(\cdot)$:

$$|\Pr[D^*(\mathsf{View}_{\mathsf{Real}}) = 1] - \Pr[D^*(\mathsf{View}_{\mathsf{Sim}}) = 1]| \geq 1/p^*(n)$$

Proof. As before, this will follow from a series of subclaims proven by fixing the randomness r of the experiment prior to the session of Π. First, observe that $\mathsf{View}[H_0]$ is trivially identically distributed to $\mathsf{View}_{\mathsf{Real}}$, since the way the receiver generates its messages and output is unchanged. Next:

Subclaim 7. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_0]|_r) = 1] - \Pr[D(\mathsf{View}[H_1]|_r) = 1]| \geq 1/p'(n)$$

Proof. We begin by analyzing the views through the first message sent by the (honest) receiver. Letting $\mathsf{View}^*_i[H_0]|_r$ denote $\mathsf{View}[H_0]|_r$ with all messages after the receiver's first message in Π removed (and respectively for H_1), we show:

Subclaim 8. There exists no $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}^*_i[H_0]|_r) = 1] - \Pr[D(\mathsf{View}^*_i[H_1]|_r) = 1]| \geq 1/p'(n)$$

Proof. This will follow from the simulation-based security of the NISC protocol π. Assuming for the sake of contradiction that there exists some $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D which distinguishes the respective truncated views with non-negligible probability $1/p'(n)$, we can construct a distinguisher D' which distinguishes between the real and simulated executions of π on the same inputs x, y, aux as those given to the session of Π.

D', given some view (m_1, m_2, out) of the messages and receiver's output from either the real interaction (between the corrupted sender and honest receiver) or the ideal interaction (between the corrupted sender and simulator $\mathcal{S}'_S$), does as follows:

- Run the experiment given by H_0 until the session of Π, using the fixed randomness r. Internally run the $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$ in place of the decommitment helper $\mathcal{H}$ throughout the experiment. (This will require time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$.)
- During Π, replace the honest receiver's first NISC message msg_1 by m_1.
- After the receiver sends their first message, run D on the resulting view and output the result.

If the given view is the real execution of the NISC protocol π, the view given to D will be identically distributed to $\mathsf{View}_i^*[H_0]|_{r||\mathsf{aux}}$; if the view is the simulated execution using $\mathcal{S}'_R$, then the view given to D will be identically distributed to $\mathsf{View}_i^*[H_1]|_{r||\mathsf{aux}}$. Hence, D' must distinguish the views of the real and simulated interactions with probability $1/p'(n)$, contradicting the definition of simulation-based security of π. □

It suffices, then, to compare the honest receiver's output between the two hybrids. In particular, notice that the verification performed in H_1 is precisely the verification of the ZK proof with the addition that $\perp$ will be returned if the malicious sender provides an invalid witness w_1 (i.e., it either provides a witness w_2 or both invalid witnesses).

So, if the ZK proof zk_2 fails to verify given the malicious sender's inputs, the receiver will return $\perp$ in both cases; if the ZK proof accepts and the sender used a statement (v, c_1, c_2) and witness $w_1 = (r_{\mathsf{NISC}}, y)$ satisfying part (a) of the language L (which, in particular, proves that c_1 is a valid commitment to w_1 and that the NISC was correctly generated with respect to the same y^* as in w_1), then both experiments will return $\perp$ if c_x is not correctly input by the sender and otherwise return $f(x, y^*)$, where y^* is the value committed to in c_1 (which, by perfect binding of $(\mathsf{Com}, \mathsf{Open})$, must be unique). Hence, there are two cases where it is possible for the outputs of the receiver to differ:

1. The malicious sender provides a proof zk_2 for a statement $(v, c_1, c_2) \notin L$ which accepts. (In this case, the commitments c_1 and c_2 to the witnesses w_1 and w_2 may be invalid.)
2. The malicious sender provides a witness $w_2 = (r_{\mathsf{NISC}}, t, z)$ proving that (v, c_1, c_2) satisfies part (b) of the language L with respect to the trapdoor t chosen by the receiver, in such a way that the NISC in H_0 outputs something besides $\perp$. (Note that H_1 will always result in output $\perp$ if the sender provides a witness w_2.)

The following two subclaims, then, complete the proof by demonstrating that the receiver's outputs are in fact statistically closely distributed between the two hybrids:

Subclaim 9. Case (1) above can occur with probability at most negligible in n during the session of Π comprising $\mathsf{View}[H_0]|_r$.

Proof. This follows from adaptive soundness of the ZK protocol $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$; specifically, assuming that case (1) does occur with some non-negligible probability $1/p''(n)$, we can use this to construct a cheating prover P^* which runs in time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$ and breaks adaptive soundness by simply running the experiment H_0 with randomness r fixed (and internally running the $T_{\mathsf{Com}}(n)$-time oracle $\mathcal{O}$ in place of the CCA helper $\mathcal{H}$) and, in the session of Π, selecting the statement (v, c_1, c_2) returned by the malicious sender and the proof zk_2. Since we assumed that, with probability $1/p''(n)$, $(v, c_1, c_2) \notin L$ and $\mathsf{ZK}_3(\mathsf{zk}_2, (v, c_1, c_2), \sigma_{\mathsf{ZK}}) = \mathsf{Accept}$, this directly implies that the cheating prover P^* will provide an accepting proof of a statement $(v, c_1, c_2) \notin L$ with non-negligible probability. This contradicts the $T_{\mathsf{ZK}}(\cdot)$-time adaptive soundness of $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$, since the cheating prover runs in time $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n) \ll T_{\mathsf{ZK}}(n)$. □

Subclaim 10. Case (2) above can occur with probability at most negligible in n during the session of Π comprising $\mathsf{View}[H_0]|_r$.

Proof. This follows from CCA security of the commitment scheme $(\mathsf{Com}, \mathsf{Open})$. Assume towards a contradiction that, in the experiment H_0, the malicious sender, with some probability $1/p''(n)$, provides a statement (v, c_1, c_2) and a witness $w_2 = (r_{\mathsf{NISC}}, t, z)$ which proves that the NISC message msg_2 was correctly generated with respect to the trapdoor t, and in addition that this trapdoor t is the same trapdoor chosen by the receiver, so that the NISC in H_0 outputs something besides $\perp$. (Notice that, if t is the *incorrect* trapdoor, the NISC in H_0 will output $\perp$, since it must receive $y = \perp$ as an input for w_2 to be valid.) Given this, we can construct a polynomial-time adversary $\mathcal{A}_{\mathsf{Com}}$ that breaks CCA security of the underlying commitment scheme, as follows:

- Run the experiment given by H_0 with the following differences:
  - Use the fixed randomness r prior to the session of Π.
  - Use CCA oracle queries in place of queries to the decommitment helper $\mathcal{H}$ for commitments using corrupted parties' identifiers.
  - Respond with $\perp$ to commitment queries using honest parties' identifiers.
  - When the trapdoor t is generated, send values t and t' (for an arbitrary $t' \neq t$) and tag (id, P_R) to obtain a commitment c^* of a randomly chosen one of the two values under the respective tag. $\mathcal{A}_{\mathsf{Com}}$ continues the experiment as before, but substitutes c^* for c_t.
  - Substitute c^* for c_2 whenever it occurs in the session of Π.
- On receiving a second message $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ from the corrupted sender, verify zk_2, and return 0 if it fails.
- Otherwise, extract a witness w_2 by running the CCA oracle on c_2. If w_2 is a tuple $(r_{\mathsf{NISC}}, t^*, z)$ such that $t^* = t$, then return 1; otherwise return 0.

Notice that c_2 (in order to verify) must be generated with the tag (id, P_S) and the receiver is honest, so the condition that the CCA oracle is never called on the tag (id, P_R) is satisfied. If c^* is a commitment of t, then the experiment is identically distributed to H_0; hence, by the assumption that case (2) occurs with probability $1/p''(n)$, $\mathcal{A}_{\mathsf{Com}}$ returns 1 with probability at least $1/p''(n)$. Otherwise, if c^* is a commitment of a random $t' \neq t$, the verification will only succeed either if w_1 and c_1 are valid or if w_2 is a decommitment of c^*, which, by perfect binding, can only

be true for some witness $(r_{\mathsf{NISC}}, t', z)$. Since $t' \neq t$, $\mathcal{A}_{\mathsf{Com}}$ will always return 0 in this case, unless the sender returns both a valid witness w_1 and a witness w_2 that does not verify but commits to t; this latter case can happen with at most probability negligible in n since the input to the sender is completely independent of t. Hence, $\mathcal{A}_{\mathsf{Com}}$ distinguishes between commitments of t and t' with probability $1/p''(n) - \nu(n)$ (for some negligible $\nu(\cdot)$), thus breaking CCA security of the commitment scheme. □

□

Subclaim 11. There exists no polynomial-time distinguisher D with oracle access to the decommitment helper $\mathcal{H}$ such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr\left[D(\mathsf{View}[H_1]|_r) = 1\right] - \Pr\left[D(\mathsf{View}[H_2]|_r) = 1\right]| \geq 1/p'(n)$$

Proof. We can prove this by using the CCA security of $(\mathsf{Com}, \mathsf{Open})$. Assume for the sake of contradiction that there exists a distinguisher D, running in polynomial time but with oracle access to $\mathcal{H}$, that distinguishes between $\mathsf{View}[H_1]|_r$ and $\mathsf{View}[H_2]|_r$ with probability $1/p'(n)$. In particular, this means that there exists some sequence r' of the receiver's first-round randomness (including the randomness used to generate the simulated NISC message msg_1) such that D distinguishes between $\mathsf{View}[H_1]|_{r||r'}$ and $\mathsf{View}[H_2]|_{r||r'}$ with probability $1/p'(n)$. We can use this to construct a polynomial-time adversary $\mathcal{A}_{\mathsf{Com}}$ which breaks the CCA security of $(\mathsf{Com}, \mathsf{Open})$. $\mathcal{A}$ will receive as non-uniform advice the first-round message msg_1 generated from the randomness r' (since it cannot run the $T_\pi(n)$-time NISC simulator), and proceeds as follows:

- Send the values $x||t$ and $0||t$ and the tag (id, P_R) to the challenger to receive a commitment c^* of a randomly chosen one of the two values.
- Run the experiment given by H_1 with the following differences:
  - Use the fixed randomness r prior to the session of Π.
  - Use the randomness r' during Π, and, rather than running the simulator $\mathcal{S}'_S$, send msg_1 as the first NISC message.
  - Use CCA oracle queries in place of queries to the decommitment helper $\mathcal{H}$ for commitments using corrupted parties' identifiers.
  - Respond with $\perp$ to commitment queries using honest parties' identifiers.
  - Substitute c^* for c_x whenever it occurs in the session of Π.
- Once the experiment finishes, run the distinguisher D on the final view and output the result.

Since R is honest, $\mathcal{A}_{\mathsf{Com}}$ never makes a CCA oracle query using the challenge commitment's tag (id, P_R). When c^* commits to $x||t$, the view given to D is identically distributed to $\mathsf{View}[H_1]|_r$; meanwhile, when c^* commits to $0||t$, the view is identically distributed to $\mathsf{View}[H_2]|_r$. Hence, if D distinguishes between $\mathsf{View}[H_1]|_r$ and $\mathsf{View}[H_2]|_r$ with probability $1/p'(n)$, $\mathcal{A}_{\mathsf{Com}}$ can distinguish between the experiments Exp_0 and Exp_1 in the CCA security game with probability $1/p'(n)$ and

Simulator $\mathcal{S}_N$

1. Generate trapdoor $t \leftarrow \{0,1\}^n$ and randomness $r_x \leftarrow \{0,1\}^*$.
2. Compute $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), 0||t; r_x)$.
3. Compute $(\mathsf{msg}_1, \sigma_{\mathsf{NISC}}) \leftarrow \mathsf{NISC}_1(1^n, (0, r_x, t))$.
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 the adversary.
6. Receive the output z from the ideal functionality $\mathcal{T}_f$ (after the honest receiver and sender have sent their respective inputs).
7. Generate randomness $r_{\mathsf{NISC}} \leftarrow \{0,1\}^*$.
8. Compute $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \bot, t, z); r_{\mathsf{NISC}})$.
9. Let $v = (\mathsf{msg}_1, \mathsf{msg}_2, c_x)$, $w_1 = (\bot, \bot)$, and $w_2 = (r_{\mathsf{NISC}}, t, z)$. Compute $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_1)$ and $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$.
10. Compute $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (v, c_1, c_2), (w_1, \bot, w_2, r_2))$.
11. Send $(\mathsf{msg}_2, \mathsf{zk}_2, c_1, c_2)$ on behalf of the sender. Send z as the receiver's final output.

Figure 6: Simulator for the case where neither party is corrupted.

without making queries to commitments with the challenge tag, contradicting the CCA security of $(\mathsf{Com}, \mathsf{Open})$. (Furthermore, since $\mathcal{A}_{\mathsf{Com}}$ is given the hard-coded simulated NISC message msg_1 instead of running the NISC simulator, it runs in polynomial time with access to the CCA oracle.) □

Subclaim 12. There exists no $T_{\mathsf{Com}}(n)\cdot\mathsf{poly}(n)$-time distinguisher D such that, for some polynomial $p'(\cdot)$ and some fixed randomness r prior to the session of Π:

$$|\Pr[D(\mathsf{View}[H_2]|_r) = 1] - \Pr[D(\mathsf{View}[H_3]|_r) = 1]| \geq 1/p'(n)$$

Proof. Identical to Subclaim 8, since only the first message changes between the two hybrids. □

Since the view of H_3 is by inspection identically distributed to the simulated experiment $\mathsf{Exp}_{\mathsf{Sim}}$, this suffices to complete the proof of the overall claim by a standard hybrid argument (i.e., if there were a D^* contradicting the claim, it would necessarily also contradict one of the above subclaims 7-12 by distinguishing the respective distributions with probability $1/3p^*(n)$).

□

5.3 Honest Receiver and Sender

Lastly, if both parties are honest in the session of Π, the simulator $\mathcal{S}$ will run $\mathcal{S}_N$ as given in Figure 6 in the experiment $\mathsf{Exp}_{\mathsf{Sim}}$. We omit the proofs for the following hybrids, as they all mirror their counterparts in the corrupted sender or receiver case (with the exception that the constructed adversaries will run the honest protocol instead of the corrupted sender or receiver).

- Let H_0 be identical to $\mathsf{Exp}_{\mathsf{Real}}$, with the exception that both parties send their respective inputs x and y to an instance of the ideal functionality $\mathcal{T}_f$ and the receiver outputs the result from $\mathcal{T}_f$.

  Views are statistically close by correctness of Π.

- Let H_1 be identical to H_0, with the exception that the sender computes the commitment c_2 as $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_2; r_2)$ (where $w_2 = (r_{\mathsf{NISC}}, t, z)$), rather than as $c_2 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_2)$.

  See Subclaim 1.

- Let H_2 be identical to H_1, with the exception that the sender computes the second ZK message as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$ rather than computing it as $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{zk}_1, (v, c_1, c_2), (w_1, r_1, w_2, \perp))$.

  See Subclaim 2.

- Let H_3 be identical to H_2, with the exception that both parties compute their respective NISC messages using the honest sender/receiver simulator $\mathcal{S}'_N$ for the underlying NISC π.

  See Subclaim 3; note that the simulator $\mathcal{S}'_N$ generates both the sender's and receiver's messages in this case.

- Let H_4 be identical to H_3, with the exception that the receiver computes the commitment to x as $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), 0||t; r_x)$ rather than $c_x = \mathsf{Com}(1^n, (\mathsf{id}, P_R), x||t; r_x)$.

  See Subclaim 11.

- Let H_5 be identical to H_4, with the exception that the sender computes the second NISC message as $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(\mathsf{msg}_1, (c_x, \perp, t, z))$—that is, using the trapdoor—rather than computing it using the simulator $\mathcal{S}'_R$ as in H_3.

  See Subclaim 4.

- Let H_6 be identical to H_5, with the exception that the sender computes the second ZK message as $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(\mathsf{zk}_1, (v, c_1, c_2), (w_1, \perp, w_2, r_2))$, where $w_1 = (\perp, \perp)$ and $w_2 = (r_{\mathsf{NISC}}, t, z)$, rather than computing it as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (v, c_1, c_2))$.

  See Subclaim 5.

- Let H_7 be identical to H_6, with the exception that the sender computes the commitment c_1 as $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), 0; r_1)$, rather than as $c_1 = \mathsf{Com}(1^n, (\mathsf{id}, P_S), w_1; r_1)$. Note that H_7 is now identical to the experiment $\mathsf{Exp}_{\mathsf{Sim}}$ where the adversary observes messages from the simulator $\mathcal{S}_N$.

  See Subclaim 6.

The above sequence of hybrids is sufficient to prove the following claim, which completes the proof of Lemma 1 and in turn Theorem 6:

Claim 3. For the case where neither party is corrupted in the session of Π, there exists no polynomial-time distinguisher D^* with oracle access to $\mathcal{H}$ such that, for some polynomial $p^*(\cdot)$:

$$|\Pr[D^*(\mathsf{View}_{\mathsf{Real}}) = 1] - \Pr[D^*(\mathsf{View}_{\mathsf{Sim}}) = 1]| \geq 1/p^*(n)$$

□

6 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:

Lemma 2. 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, 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 4. $(\mathsf{Com}, \mathsf{Open})$ satisfies perfect binding.

Proof. 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. Then assume for the sake of contradiction that there exists a commitment c and two pairs $(x, (p, r))$ and $(x', (p', r'))$ such that $x \neq x'$ but $\mathsf{Open}(c, x, (p, r)) = \mathsf{Accept}$ and $\mathsf{Open}(c, x', (p', r')) = \mathsf{Accept}$ both with non-zero probability.

First, note that this implies that c is both a correctly generated commitment to x under (p, r) and a correctly generated commitment to x' under (p', r'), as otherwise the respective opening will return Reject with probability 1. And, given that the commitments are correctly generated (and thus honestly generated first-round messages of Π), correctness[6] of Π implies that, in fact, $\mathsf{Open}(c, x, (p, r)) = \mathsf{Accept}$ and $\mathsf{Open}(c, x', (p', r')) = \mathsf{Accept}$ both with probability 1; this follows since both of these are the result of running the honest protocol Π with the valid first message c, and thus must return the correct result from Π (either Accept or Reject) with probability 1, which must be Accept due to our earlier assumption that Open returns Accept with non-zero probability on both of these commitments.

Towards our contradiction, we examine the security of Π by constructing an environment $\mathcal{Z}$ for any sufficiently large security parameter $n \in \mathbb{N}$ (i.e., any n such that $x||p$ and $x'||p'$ are valid inputs). Letting id be the identity corresponding to c, $\mathcal{Z}$, 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 $\pi_3(m)|_r$.

Considering the real execution of the above, on input $x^* = x||p$ for the sender, we notice that by perfect correctness the output must be 1 with probability 1, whereas if instead we provide input $x^* \neq x||p$ the output must be 0 with probability 1. So, by the security of Π, it must be the case that the final output of the ideal execution is 1 except with negligible probability when the sender's input is $x||p$ and 0 except with negligible probability when the input is anything else. However, in the ideal version of the execution, notice that the only input to the simulator $\mathcal{S}$ that is dependent on the sender's input is the output from the ideal functionality $\mathcal{T}_f$.

We can in fact use this to deduce that the input extracted by the simulator $\mathcal{S}$ from c and sent to the ideal functionality on behalf of the corrupted sender is $x||p$ with overwhelming probability. If not, then there exists a proper subset X of all x^* not containing $x||p$ such that $\mathcal{S}$ extracts a member of X with some non-negligible probability $1/p(n)$. But this means that, comparing the case where the sender's input is $x||p$ to a case where the sender's input is $x^* \neq x||p$ but also is a non-member of X, the inputs to the simulator are identically distributed with non-negligible probability (i.e., when $\mathcal{T}_f$ returns 0 because a member of X was selected), and thus it is impossible for the output

[6]Note that our definition specifies *perfect* correctness, as is indeed satisfied by our construction in Theorem 6.

of the ideal interaction in the former case to be 1 except with negligible probability and the output of the ideal interaction in the latter case to be 0 except with negligible probability as is required for security, as for that to be true the distributions would have to share only a negligible fraction of probability mass.

This in itself is not a contradiction; 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 sender is $x'||p'$ with overwhelming probability. Clearly, this cannot be true simultaneously with the above fact; thus, by contradiction, $(\mathsf{Com}, \mathsf{Open})$ must satisfy perfect binding. □

Claim 5. $(\mathsf{Com}, \mathsf{Open})$ 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 13. 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[7], and with the 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

[7] 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.

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, $\mathcal{Z}$ receives as non-uniform advice the i^{th} query c and (padded) decommitment $x||p$ (which can be $\perp$ if c is an invalid commitment), which are deterministic given fixed randomness r and the responses from the true CCA oracle to the first $i-1$ queries, and does 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 oracle queries 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 $x'||p'$ with $x' \neq x$ 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. □

We also show the following, which together with the previous claim will provide a contradiction:

Subclaim 14. $(\mathsf{Com}, \mathsf{Open})$ 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.

Proof. First, let us consider an "ideal" commitment scheme $(\mathsf{Com}', \mathsf{Open}')$, defined identically to $(\mathsf{Com}, \mathsf{Open})$ except that, rather than using the protocol Π, $(\mathsf{Com}', \mathsf{Open}')$ runs the idealized version of the protocol, which we shall call Π', where the sender's and receiver's inputs in π_1 and π_2 are sent to an ideal functionality $\mathcal{T}_f$ that computes and outputs the result of the function f (i.e., the equality function) and messages are generated independently of the respective inputs by using the simulator $\mathcal{S}$.

We claim that $(\mathsf{Com}', \mathsf{Open}')$ trivially satisfies hiding, even against unbounded-time adversaries; this follows since, given a randomly generated commitment c to some value x, the respective receiver input to the idealized protocol Π' is $x||p$ for some random padding $p \leftarrow \{0,1\}^n$. Furthermore, since the commitment c is given by the simulated first-round message of the idealized protocol, it is generated completely independently of x or p. So, given an adversary $\mathcal{A}$ which picks two values

x_0 and x_1 and receives a commitment to a random one of the two, the only way $\mathcal{A}$ can receive any information about the committed value is by attempting to open the commitment—i.e., by interacting with the ideal functionality. However, since the receiver input contains not only the value x committed but also the random padding p, the ideal functionality will return 0 unless the adversary manages to guess both x and p correctly. And since the ideal functionality computes f only once, and the commitment c by construction is independent of $x||p$, $\mathcal{A}$ can guess p with probability at most 2^{-n}. The remainder of the time, it receives inputs which are entirely independent of the committed value, meaning that it clearly cannot distinguish the two commitments with non-negligible probability even if given unbounded time.

This is precisely what we need to prove the claim, since the remainder follows by the relativized universal composition theorem and the $\mathcal{H}$-EUC security of Π. Consider the following protocol G defining the hiding security game between the adversary and the challenger:

- The adversary, given input 1^n and some auxiliary input z, selects values x_0 and x_1 and sends them to the challenger.
- The challenger, given input 1^n, session identifier id, and $b \in \{0,1\}$, receives the input from the adversary and generates a commitment $c \leftarrow \mathsf{Com}(1^n, \mathsf{id}, x_b)$, which it sends to the adversary.
- The adversary receives c and produces an arbitrary output.

Let G' be defined identically to G except that G' will use the idealized Com' in place of Com. G' is clearly a $\mathcal{T}_f$-hybrid protocol (recall that $\mathcal{T}_f$ is the ideal equality functionality implemented by Π), and Π $\mathcal{H}$-EUC-realizes $\mathcal{T}_f$ by assumption; therefore, G, which is simply the composed protocol where Π replaces $\mathcal{T}_f$, $\mathcal{H}$-EUC-emulates G'. This means that, given any polynomial-time adversary $\mathcal{A}$ in the hiding security game—even if $\mathcal{A}$ has access to the superpolynomial-time helper $\mathcal{H}$—and any inputs $1^n, \mathsf{id}, z, b$, no polynomial-time distinguisher D can distinguish the distribution of the adversary's output in the ideal execution of G' from the distribution of the adversary's output in the real execution of G. That is, if for $b \in \{0,1\}$ we let $\mathsf{Exp}_b(\mathcal{A}, n, z)$ be the adversary's output distribution in G with inputs $1^n, z, b$, as in Definition 9, and $\mathsf{Exp}'_b(\mathcal{A}, n, z)$ defined respectively for G', we are guaranteed that there exists negligible $\nu(\cdot)$ such that, for any polynomial-time distinguisher D:

$$|\Pr[D(\mathsf{Exp}_b(\mathcal{A}, n, z)) = 1] - \Pr[D(\mathsf{Exp}'_b(\mathcal{A}, n, z)) = 1]| \leq \nu(n)$$

But hiding in the ideal world provides that:

$$|\Pr[D(\mathsf{Exp}'_0(\mathcal{A}, n, z)) = 1] - \Pr[D(\mathsf{Exp}'_1(\mathcal{A}, n, z)) = 1]| \leq 2^{-n}$$

and so we have:

$$|\Pr[D(\mathsf{Exp}_0(\mathcal{A}, n, z)) = 1] - \Pr[D(\mathsf{Exp}_1(\mathcal{A}, n, z)) = 1]| \leq 2\nu(n) + 2^{-n}$$

which is sufficient to prove hiding as desired. Notice, however, that $\mathcal{H}$-EUC security requires the environment $\mathcal{Z}$ to query the helper $\mathcal{H}$ only on behalf of corrupted parties. Since the challenge commitment c, given its tag id, requires an instance of Π to be started with honest parties and session identifier id, the above holds only if the adversary $\mathcal{A}$ does not query $\mathcal{H}$ with the same session identifier. □

So, given an adversary $\mathcal{A}$ that contradicts weak CCA security using polynomial time and oracle access to the CCA oracle $\mathcal{O}^*$, Subclaim 13 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 14, 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] 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)

[2] 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)

[3] 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)

[4] 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)

[5] Beaver, D.: Foundations of secure interactive computing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 377–391. Springer, Heidelberg (Aug 1992)

[6] 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)

[7] 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)

[8] 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)

[9] 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)

[10] Brassard, G., Chaum, D., Crépeau, C.: Minimum Disclosure Proofs of Knowledge. J. Comput. Syst. Sci. 37(2), 156–189 (Oct 1988)

[11] Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (Jan 2000)

[12] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001)

[13] 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)

[14] 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)

[15] 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)

[16] Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (1998), manuscript

[17] 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)

[18] Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. In: 30th ACM STOC. pp. 409–418. ACM Press (May 1998)

[19] Feige, U.: Alternative Models for Zero-Knowledge Interactive Proofs. Ph.D. Thesis, Weizmann Institute of Science (1990)

[20] 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)

[21] 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)

[22] 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)

[23] 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)

[24] Goldreich, O.: Foundations of Cryptography: Basic Tools, vol. 1. Cambridge University Press, Cambridge, UK (2001)

[25] 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)

[26] Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7(1), 1–32 (Dec 1994)

[27] 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)

[28] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1), 186–208 (1989)

[29] 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)

[30] 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)

[31] 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)

[32] 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)

[33] 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)

[34] 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)

[35] 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)

[36] Lindell, Y.: Lower bounds for concurrent self composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (Feb 2004)

[37] 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)

[38] Micali, S., Rogaway, P.: Secure computation (abstract). In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 392–404. Springer, Heidelberg (Aug 1992)

[39] 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)

[40] 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)

[41] 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)

[42] Pass, R., Rosen, A.: Concurrent non-malleable commitments. In: 46th FOCS. pp. 563–572. IEEE Computer Society Press (Oct 2005)

[43] 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)

[44] 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)

[45] Schröder, D., Unruh, D.: Round optimal blind signatures. Cryptology ePrint Archive, Report 2011/264 (2011), `https://eprint.iacr.org/2011/264`

[46] Yao, A.C.C.: Protocols for secure computations (extended abstract). In: 23rd FOCS. pp. 160–164. IEEE Computer Society Press (Nov 1982)

[47] Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS. pp. 160–164 (1982)

A 2-round SPS-ZK from SPS-NISC

We restate Theorem 5:

Theorem 8 (Theorem 5 restated)**.** 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 stand-alone 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.

Proof. Given an NP language $\mathcal{L} \subseteq \{0,1\}^*$ with witness relation $R_{\mathcal{L}}$, we define a zero-knowledge interactive argument protocol for $\mathcal{L}$ as follows:

Let $\Pi = (\pi_1, \pi_2, \pi_3)$ be a protocol for stand-alone secure non-interactive computation of the functionality which, on inputs $\perp$ from the receiver and (x, w) from the sender, returns (x, Accept) if $(x, w) \in \mathcal{L}$ and (x, Reject) otherwise. (Note that, since $\mathcal{L}$ is an NP language, this functionality is guaranteed to be polynomial-time computable.)

To define our zero-knowledge argument ZK, the prover and verifier, on common input x and the prover's witness w, will simply run Π with the verifier acting as the receiver and the prover acting as the sender, and the prover will input (x, w). The verifier will accept if Π outputs (x', Accept) with $x = x'$, and reject otherwise.

Completeness follows immediately from the correctness of Π, as if $(x, w) \in \mathcal{L}$ then the functionality f will always output (x, Accept). So it suffices to prove adaptive soundness and simulatability.

To prove simulatability, assume for the sake of contradiction that ZK is not simulatable—that is, there exists some cheating verifier V^* such that, for any $T'(\cdot)$-time simulator Sim, there exists polynomial $p(\cdot)$, distinguisher D, statement $x \in \mathcal{L}$, witness $w \in R_{\mathcal{L}}(x)$, and auxiliary input z such that:

$$|\Pr\left[D(x, \langle P(w), V^*(z)\rangle(x)) = 1\right] - \Pr\left[D(x, \mathsf{Sim}(x, z)) = 1\right]| \geq 1/p(|x|)$$

By stand-alone security of Π given the cheating verifier V^* as the adversary $\mathcal{A}$, there exists a $T'(\cdot)$-time simulator $\mathcal{S}$ such that, for any non-uniform polynomial-time environment $\mathcal{Z}$ and non-uniform polynomial-time distinguisher D', there exists negligible $\nu(\cdot)$ such that, for any $n \in \mathbb{N}$:

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

If we define $\mathcal{Z}$ to be the environment which invokes a single instance of Π with an adversarial receiver and inputs $\perp$ and (x, w), if we define a simulator Sim which runs $\mathsf{Exec}'_{\Pi', \mathcal{S}, \mathcal{Z}}(1^n, z)$ (i.e., invokes the environment $\mathcal{Z}$ using the simulated execution rather than the real one), if we define a distinguisher D' which determines x from the view of Π (note that x is part of the output) and runs D on x and its own input, and if we let $n = |x|$, then we see that:

$$D'(\mathsf{Exec}'_{\Pi, V^*, \mathcal{Z}}(1^n, z)) = D(x, \langle P(w), V^*(z)\rangle(x))$$

and

$$D'(\mathsf{Exec}'_{\Pi', \mathcal{S}, \mathcal{Z}}(1^n, z)) = D(x, \mathsf{Sim}(x, z))$$

Of course, this implies, by our assumption that simulatability fails, that:

$$|\Pr\left[D'(\mathsf{Exec}'_{\Pi, V^*, \mathcal{Z}}(1^n, z)) = 1\right] - \Pr\left[D'(\mathsf{Exec}'_{\Pi', \mathcal{S}, \mathcal{Z}}(1^n, z)) = 1\right]| \geq 1/p(n)$$

contradicting stand-alone security of Π.

To prove soundness, once again assume the opposite for the sake of contradict-ion—that is, for some cheating prover P^*, polynomial $p(\cdot)$, and $n \in \mathbb{N}$, there is at least a $1/p(n)$ probability that the prover, when selecting a false statement, can convince the verifier, or, in other words, that Π will output (x^*, Accept) given the assumption that P^* (adaptively) selects some $x^* \notin \mathcal{L}$.

We show that this contradicts stand-alone security of Π, as follows. Consider an environment $\mathcal{Z}$ which runs an instance of Π with P^* as the corrupted sender. In the real execution, we know by our assumption that this interaction must return (x^*, Accept) with probability $1/p(n)$. However, consider the simulated execution where, given the simulator $\mathcal{S}$ from the definition of stand-alone

security, $\mathcal{S}$ sends a simulated first message to P^*, receives a response, and produces an input (x', w) to send to the ideal functionality $\mathcal{T}_f$, which will produce the final output. Because the prover P^* must select a statement $x^* \notin \mathcal{L}$, it follows that this interaction can produce $(x', w) \in R_{\mathcal{L}}$, and thus that the simulated interaction (which runs the actual functionality for the witness relation) can return (x^*, Accept), with at most probability negligible in n, thus allowing the real and simulated interactions to be distinguished by a distinguisher D that simply returns whether the interaction accepts.

□