Succinct Non-Interactive Secure Computation[*]

Andrew Morgan
Cornell University
asmorgan@cs.cornell.edu

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

Antigoni Polychroniadou[‡]
J.P. Morgan AI Research
antigonipoly@gmail.com

February 20, 2020

Abstract

We present the first *maliciously secure* protocol for *succinct non-interactive secure two-party computation* (SNISC): Each player sends just a single message whose length is (essentially) independent of the running time of the function to be computed. The protocol does not require any trusted setup, satisfies superpolynomial-time simulation-based security (SPS), and is based on (subexponential) security of the Learning With Errors (LWE) assumption. We do *not* rely on SNARKs or "knowledge of exponent"-type assumptions.

Since the protocol is non-interactive, the relaxation to SPS security is needed, as standard polynomial-time simulation is impossible; however, a slight variant of our main protocol yields a SNISC with polynomial-time simulation in the CRS model.

[*]A preliminary version of this paper will appear in the proceedings of EUROCRYPT 2020; this is the full version.

[†]Supported in part by NSF Award SATC-1704788, NSF Award RI-1703846, and AFOSR Award FA9550-18-1-0267. This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via 2019-19-020700006. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

[‡]This paper was prepared in part for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful. © 2020 JPMorgan Chase & Co. All rights reserved

1 Introduction

Protocols for *secure two-party computation* (2PC) allow two parties to compute any function (f) of their private inputs (x and y) without revealing anything more than the output $f(x, y)$ of the function. Since their introduction by Yao [Yao82] and Goldreich, Micali and Wigderson [GMW87], they have become one of the most central tools in modern cryptography. In this work, our focus is on 2PC in a setting with a *non-interactivity* requirement: each player sends just a *single* message. The first player—typically referred to as the *receiver* (or R)—computes some message m_1 based on its input x and sends m_1 to the second player. The second player—referred to as the *sender* (S)—next computes a response m_2 (based on its input y and the message m_1 it received) and sends it back to the receiver. Upon receiving the response m_2, the receiver can finally compute and output $f(x, y)$. (Note that in such a non-interactive scenario, it is essential that only the receiver obtains the output—in other words, that the functionality is "one-sided"; otherwise, since the protocol only has two rounds, the sender will be able to compute the output given only m_1, meaning that it could obtain $f(x, y^*)$ on any number of inputs y^* of its choice.)

SNISC: Succinct Non-Interactive Secure Computation. As far as we know, this notion of non-interactive 2PC was first formally studied in [IKO$^+$11] under the name *non-interactive secure computation (NISC)*; however, informal versions of it became popular in connection with Gentry's breakthrough result on *fully homomorphic encryption* (FHE) [Gen09]. One of the original applications of FHE was the *private outsourcing* of some computation to a remote party: for instance, consider a scenario where a client (the receiver) has some secret input x and wishes a powerful server (the sender) to compute some potentially time-consuming function f on x (and potentially another input y belonging to the server). Using FHE, the client/receiver simply lets m_1 be an FHE encryption of x; the server/sender can next use homomorphic evaluation to obtain an encryption m_2 of $f(x, y)$ to send back, which can be decrypted by the client/receiver. Indeed, an FHE scheme not only directly yields a NISC, but it also yields a *succinct NISC (SNISC)*—where both the communication complexity of the protocol and the running time of an honest receiver are "essentially" independent of the running time of f. More formally, we define a SNISC as a NISC where the communication complexity and receiver running time depend only on the length of the inputs and outputs, and *polylogarithmically* on the running time of the function f to be computed (where we assume that f is given as a Turing machine).

The problem with this folklore approach towards private outsourcing/succinct NISC is that using FHE alone only satisfies *semi-honest security*, as opposed to fully *malicious security*. For instance, a malicious sender could decide to compute any other function of its choice instead of the correct f! Of course, we could always extend the protocol using ZK-SNARKs (succinct non-interactive arguments of knowledge) [Mic94, Gro10, BCCT13, BCI$^+$13, GGPR13] to prove correctness of the messages m_1 and m_2, but doing so comes at a cost. First, we now need to assume some trusted setup, such as a *common reference string* (CRS). Additionally, all known constructions of SNARKs are based on knowledge- or extractability-type assumptions, which in general are known to be problematic with respect to arbitrary auxiliary input [BCPR14, BP15].[1] Thus, the question as to whether succinct non-interactive secure computation with *malicious* security is possible in

[1]Finally, even forgetting about the issues with extractability assumptions, formalizing this approach requires dealing with some subtle issues, which we will discuss later on. Works where this has been done (in the orthogonal setting of "laconic" function evaluation) include [CDG$^+$17, QWW18].

the plain model remains open:

> *Does there exist a succinct non-interactive secure computation protocol without any trusted setup (and without using extractability assumptions)?*

NISC protocols in models *with* trusted setup have been extensively studied. There exist known constructions of NISC in the OT-hybrid model [IKO$^+$11], in the CRS model based on cut-and-choose [AMPR14, MR17], assuming stateful [GIS$^+$10] and stateless [HPV16, BJOV18] tamper-proof hardware tokens, and in the global random oracle model [CJS14]. As far as we know, none of the above protocols are succinct.

The plain model, however, presents additional complications: Goldreich-Oren's [GO94] classic impossibility result for two-round zero-knowledge proofs immediately shows that even a non-succinct (let alone succinct) NISC with malicious security cannot satisfy the standard *polynomial-time* simulation-based notion of security.[2] Thus, to get *any* NISC, let alone a succinct one, we need to use some relaxed notion of simulatability for the definition of secure computation. *Superpolynomial-time simulation-based security* (SPS) [Pas03, PS04] has emerged as the standard relaxation of simulation-based security: under SPS security, the attacker is restricted to be a non-uniform polynomial time algorithm, but the simulator (in the definition of secure computation) is allowed to run in (slightly) superpolynomial time (e.g., in quasi-polynomial time). Non-succinct NISC protocols with SPS simulation are known under various standard assumptions [Pas03, SU11, BGI$^+$17]. Most notably, the work of [BGI$^+$17] constructs a maliciously secure (non-succinct) NISC with quasi-polynomial simulation in the plain model which can securely compute any functionality based on the subexponential security of various standard hardness assumptions; we return to this result in more detail later on. However, all previous works only construct NISC protocols that are non-succinct.

Towards achieving succinctness for NISC, a very recent work by Brakerski and Kalai [BK18] takes us a step on the way: they focus on a notion of "private delegation" where the receiver's/client's input x is publicly known (and thus does not need to be kept hidden) but the input y of the sender/server is considered private. The authors present a delegation protocol that achieves *witness indistinguishability (WI)* for the sender—as shown in [Pas03], WI is a strict relaxation of SPS security.[3] While their protocol achieves the desired notion of succinctness, it still falls short of the goal of producing a succinct NISC protocol due to the fact that its only considers privacy for one of the players (namely, the sender); this significantly simplifies the problem. Additionally, their notion of privacy (witness indistinguishability) is also weaker than what we are aiming to achieve (i.e., simulation-based SPS security).

1.1 Our Results

In this work, we provide an affirmative answer to the above question, presenting the first SNISC for general functionalities. Our protocol is in the plain model (i.e., no trusted setup), and we do not rely on any extractability-based assumptions.

[2] Furthermore, if we restrict to black-box simulation, [KO04] proved that four rounds are necessary and sufficient for secure one-sided 2PC in the plain model.

[3] In the context of interactive proofs, WI is equivalent to a relaxed form of SPS security where the simulator's running time is unbounded (as opposed to some "small" superpolynomial time).

Theorem 1 (Informally stated). *Assuming subexponential security of the LWE assumption, there exists a maliciously SPS-secure SNISC for any efficient functionality. Furthermore, the simulator of the protocol runs in quasi-polynomial time.*

Our protocol relies on three primitives:

- A (leveled) FHE scheme [Gen09] with quasi-polynomial security. For our purposes, we additionally require the FHE to satisfy *perfect correctness*. Such schemes can be based on the (quasi-polynomial security of the) LWE (Learning With Errors) assumption [Reg05], as shown in [BGV12, GKP$^+$13].
- A (non-private) delegation scheme for polynomial time computations with quasi-polynomial security. For our purpose, we require a scheme that satisfies *perfect completeness* and allows the sender to adaptively choose the functionality (i.e., we need what is referred to as an "adaptive delegation scheme"). Such schemes can in fact be based on the above notion of quasi-polynomial FHE, and hence in turn on the quasi-polynomial security of the LWE assumption [BHK17].
- A (non-succinct) SPS-secure NISC for general functionalities f with a quasi-polynomial simulator. Such a scheme exists based on the existence of a subexponentially-secure "weak oblivious transfer" protocol[4] [BGI$^+$17][5], which in turn can be based on the subexponential security of any one of the DDH [NP01], Quadratic Residuosity, or N^{th} Residuosity [HK12] assumptions, or (as shown in [BD18]) on subexponential security of the LWE assumption.

More precisely, if the underlying NISC protocol has a $T(n) \cdot \mathsf{poly}(n)$-time simulator, and if all the other primitives are secure against $T(n) \cdot \mathsf{poly}(n)$ time attackers, the final protocol is secure and has a $T(n) \cdot \mathsf{poly}(n)$-time simulator:

Theorem 2 (Informally stated). *Assuming the existence of a $T(n)$-time simulatable NISC protocol, a subexponentially sound adaptive delegation scheme for polynomial-time computations with perfect completeness, and a subexponentially secure leveled FHE scheme with perfect correctness, there exists $T(n) \cdot \mathsf{poly}(n)$-time simulatable SNISC for any efficient functionality.*

As a corollary, we can directly instantiate our protocol using a NISC with polynomial-time simulation in the CRS model (see Appendix A) based on a two-round universally composable OT protocol (in the CRS model), which [PVW08] shows can be based on the polynomial security of LWE. Hence:

Corollary 1 (Informally stated). *Assuming the polynomial security of the LWE assumption, there exists a maliciously-secure SNISC (with a polynomial-time simulator) in the CRS model for any efficient functionality.*

[4]Roughly speaking, a weak oblivious transfer protocol is an OT protocol that satisfies SPS-security against a malicious receiver, but only indistinguishability-based ("game-based") security against a malicious sender.

[5]While [BGI$^+$17] claim a construction of SPS NISC from just the existence of a weak OT protocol, their security proof additionally relies on the existence of an *onto* one-way function. As far as we know, onto one-way functions are not known based on the existence of Weak OT. Consequently, in Appendix A we present a variant of their protocol that dispenses of this additional assumption.

1.2 Technical Overview

At a high level, our approach begins with the semi-honestly secure approach of using FHE (which we detailed in the introduction) and attempts to compile it to become secure with respect to malicious attackers. Instead of using ZK-SNARKs (which rely on non-standard assumptions and trusted setup), we will instead use an adaptive delegation scheme and a non-succinct NISC. For our approach to work, we will strongly rely on *perfect correctness/completeness* properties of both the FHE and the delegation scheme; as far as we know, perfect correctness of these types of primitives has not previously been used to enable applications (where the goal itself isn't perfect correctness).[6]. Despite this, though, recent constructions (or slight variants) of both FHE and delegation protocols fortunately do provide these guarantees.

Adaptive Delegation: A Starting Point. To explain the approach, we shall start from a (flawed) candidate which simply combines an FHE scheme and an adaptive delegation scheme. In an adaptive delegation scheme (as given in [BHK17]), a verifier generates a public/secret key-pair $(\mathsf{pk}, \mathsf{sk})$ and sends pk to the prover. The prover next picks some statement $\tilde{x}$ and function g, computes the output $\tilde{y} = g(\tilde{x})$, and produces a "short" proof π of the validity of the statement that $\tilde{y} = g(\tilde{x})$. The prover finally sends $(\tilde{x}, g, \tilde{y}, \pi)$ to the verifier, who can use its secret key sk to check the validity of the proof. We will rely on an adaptive delegation scheme satisfying *perfect completeness*—that is, for all public keys in the range of the key generation algorithm, the prover can convince the verifier with probability 1.

The candidate SNISC leverages delegation to "outsource" the computation of the homomorphic evaluation to the sender: specifically, the receiver first generates a public/secret key-pair $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}})$ for the FHE, encrypts its input x using the FHE (obtaining a ciphertext ct_x), generates a public/secret key pair $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}})$ for the delegation scheme, and finally sends $(\mathsf{ct}_x, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}})$ to the sender. The sender in turn encrypts its input y, obtaining a ciphertext ct_y; next, it lets g be the function for homomorphically evaluating f on two ciphertexts, computes $g(\mathsf{ct}_x, \mathsf{ct}_y)$ (i.e., homomorphically evaluates f on ct_x and ct_y) to obtain a ciphertext $\mathsf{ct}_{\mathsf{out}}$, and computes a delegation proof π (with respect to $\mathsf{pk}_{\mathsf{Del}}$) of the validity of the computation of g. Finally, the sender sends $(\mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi)$ to the receiver, who verifies the proof and, if the proof is accepting, decrypts $\mathsf{ct}_{\mathsf{out}}$ and outputs it.

Intuitively, this approach hides the input x of the receiver, but clearly fails to hide the input y of the sender, as the receiver can simply decrypt ct_y to obtain y. So, rather than providing ct_y and π in the clear (as even just the proof π could leak things about ct_y), we instead use the (non-succinct) NISC to run the verification procedure of the delegation scheme. That is, we can add to the protocol a NISC instance where the receiver inputs $\mathsf{sk}_{\mathsf{Del}}$, the sender inputs $\mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi$, and the functionality runs the verification algorithm for the delegation scheme, outputting either $\perp$ if verification fails or, otherwise, $\mathsf{ct}_{\mathsf{out}}$ (which can be decrypted by the receiver).

Input Independence: Leveraging Perfect Correctness of FHE. The above approach intuitively hides the inputs of both players, and also ensures that the function is computed correctly. But there are many problems with it. For instance, while it guarantees that the sender does not learn the receiver's input x, it does *not* guarantee "input independence", or that the sender's input

[6]The only work we are aware that uses perfect correctness of a FHE is a very recent work [AEKP19] which uses perfectly correct FHE as a tool to get perfectly correct iO.

does not depend on the receiver's somehow: for instance, the sender can easily maul ct_x into, say, an encryption ct_y of $x+1$ and use that as its input. On a more technical level, simulation-based security requires the simulator to be able to extract the inputs of malicious players, but it is not clear how this can be done here—in fact, a simulator *cannot* extract the sender's input y due to the above malleability attack.

To overcome this issue, we again leverage the non-succinct NISC to enable extractability: we add x and the randomness, r_x, needed to generate ct_x as an input from the receiver, and we add ct_x (i.e., the ciphertext obtained from the receiver), y, and the randomness needed to generate ct_y as input from the sender. The functionality additionally checks that the ciphertexts $\mathsf{ct}_x, \mathsf{ct}_y$ respectively are valid encryptions of the inputs x, y using the given randomness. (It is actually essential that the *sender* includes the ciphertext ct_x from the receiver as part of its input, as opposed to having the receiver input it, as otherwise we could not guarantee that the receiver is sending the same ciphertext to the sender as it is inputting to the NISC). If we have perfect correctness for the underlying FHE scheme with respect to the public-keys selected by the receiver, this approach guarantees that we can correctly extract the inputs of the players. The reason that we need perfect correctness is that the NISC only guarantees that the ciphertexts have been honestly generated using *some* randomness, but we have no guarantees that the randomness is honestly generated. Perfect correctness ensures that all randomness is "good" and will result in a "well-formed" ciphertext on which homomorphic computation, and subsequently decryption, will always lead to the correct output.

Dealing with a Malicious Receiver: Interactive Witness Encryption and Perfectly Correct Delegation. While the above protocol suffices to deal with a malicious sender (although, as we shall discuss later on, even this is not trivial due to the potential for "spooky interactions" [DLN+04]), it still does not allow us to deal with a malicious receiver. The problem is that the receiver could send invalid public keys, either for the FHE or for the delegation scheme. For instance, if the public key for the FHE is invalid, perfect correctness may no longer hold, and we may not be able to extract a correct input for the receiver. Likewise, if the public key for the delegation scheme is invalid, we will not be able to determine whether the verification algorithm of the delegation scheme will be accepting, and thus cannot carry out a simulation. Typically, dealing with a malicious receiver would require adding a zero-knowledge proof of well-formedness of its messages; however, given that the receiver is sending the first message, this seems problematic since, even with SPS-security, one-message ZK is impossible (with respect to non-uniform attackers [Pas03, BP04]).

To explain our solution to this problem, let us first assume that we have access to a *witness encryption scheme* [GGSW13]. Recall that a witness encryption scheme enables encrypting a message m with a statement $\tilde{x}$ so that anyone having a witness w to $\tilde{x}$ can decrypt the message; if the statement is false, however, the encryption scheme conceals the message m. If we had access to such a witness encryption scheme, we could have the functionality in the NISC compute a witness encryption of $\mathsf{ct}_{\mathsf{out}}$ with the statement being that the public keys have been correctly generated. This method ensures that the receiver does not get any meaningful output unless it actually generated the public keys correctly. Of course, it may still use "bad" randomness—we can only verify that the public keys are in the range of the key generating function. But, if the delegation scheme *also* satisfies a "perfect correctness" property (specifically, both correctness of the computation and *perfect completeness* of the generated proof), this enables us to simulate the verification of the delegation scheme (as once again, in this case, perfect correctness guarantees that there is no "bad" randomness).

We still have an issue: perfect correctness of the FHE will ensure that the decryption of the output is correct, but we also need to ensure that we can simulate the ciphertext output by the NISC. While this can be handled using an FHE satisfying an appropriate rerandomizability/simulatability property (also with respect to maliciously selected ciphertext), doing so introduces additional complications. Furthermore, while we motivated the above modification using witness encryption, currently known constructions of witness encryption rely on non-standard, and less understood, hardness assumptions; as such, we would like to altogether avoid using it as an underlying primitive.

So, to circumvent the use of witness encryption—while at the same time ensuring that the output of the NISC is simulatable—we realize that in our context, it in fact suffices to use a *two-round version of witness encryption*, where the receiver of the encryption chooses the statement and can first send a message corresponding to the statement. And such a non-interactive version of witness encryption can be readily implemented using a NISC! As we are already running an instance of a NISC, we can simply have the NISC also implement this interactive witness encryption. More precisely, we now additionally require the receiver to provide its witness—i.e., the randomness for the key generation algorithms—as an input to the NISC, while the sender additionally provides the public keys $\mathsf{pk}_{\mathsf{FHE}}$ and $\mathsf{pk}_{\mathsf{Del}}$ which it receives. The functionality will now only release the output $\mathsf{ct}_{\mathsf{out}}$ if it verifies that the keys input by the sender are correctly generated from the respective randomness input by the receiver. Better still, since the randomness used to generate the public/secret key-pair is now an input to the functionality, the functionality can *also* recover the secret key for the FHE, and next also decrypt $\mathsf{ct}_{\mathsf{out}}$ and simply output plain text corresponding to $\mathsf{ct}_{\mathsf{out}}$. This prevents the need for rerandomizing $\mathsf{ct}_{\mathsf{out}}$, since it is now internal to the NISC instance (and is no longer output). With all of the above modifications, we can now prove that the protocol satisfies SPS security.

The Final Protocol. For clarity, let us summarize the final protocol.

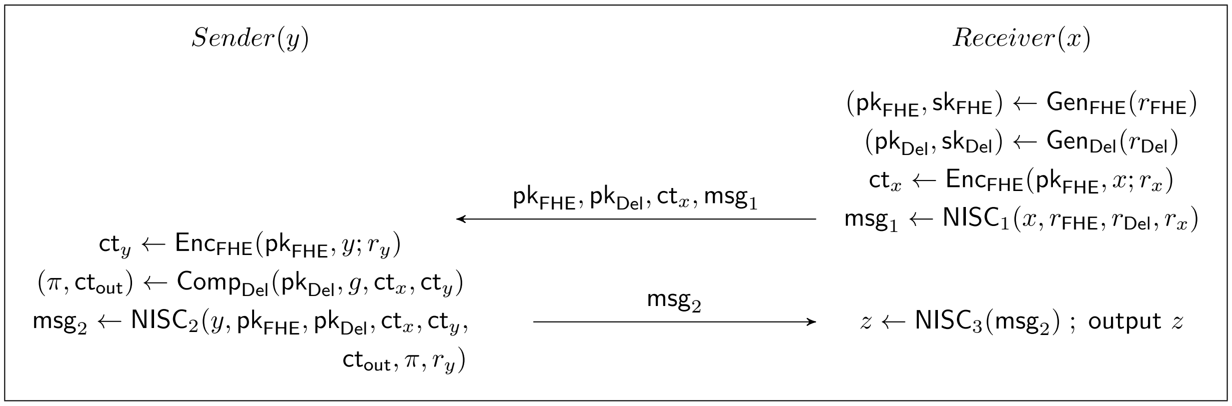
- The Receiver generates $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}})$ and $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}})$ using randomness r_{FHE} and r_{Del} (respectively) and generates an encryption ct_x of its input x using randomness r_x. It then sends $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ and the first message msg_1 of a NISC using the input $x' = (x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_x)$ (for a functionality to be specified shortly).

- The Sender, upon receiving $\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{msg}_1$ generates an encryption ct_y of its input y using randomness r_y, applies the homomorphic evaluation of f to ct_x and ct_y to obtain a ciphertext $\mathsf{ct}_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}_y)$, generates a proof π using the delegation scheme (w.r.t. $\mathsf{pk}_{\mathsf{Del}}$) of the correctness of the computation that $\mathsf{ct}_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}_y)$, and finally sends the second message msg_2 of the NISC using the input $y' = (y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi, r_y)$.

- Finally, the receiver, upon getting msg_2, computes the output z of the NISC protocol and outputs it.

- The functionality computed by the NISC on input $x' = (x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_x)$ and $y' = (y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi, r_y)$ does the following: it checks that:

  1. the public keys $\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}$ were respectively generated using randomness $r_{\mathsf{FHE}}, r_{\mathsf{Del}}$;



$Sender(y)$
$Receiver(x)$
$(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}}) \leftarrow \mathsf{Gen}_{\mathsf{FHE}}(r_{\mathsf{FHE}})$
$(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) \leftarrow \mathsf{Gen}_{\mathsf{Del}}(r_{\mathsf{Del}})$
$\mathsf{ct}_x \leftarrow \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_x)$
$\mathsf{msg}_1 \leftarrow \mathsf{NISC}_1(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_x)$
$\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1$
$\mathsf{ct}_y \leftarrow \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y; r_y)$
$(\pi, \mathsf{ct}_{\mathsf{out}}) \leftarrow \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$
$\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi, r_y)$
msg_2
$z \leftarrow \mathsf{NISC}_3(\mathsf{msg}_2)$; output z

Figure 1: The final SNISC protocol. $(\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ denotes the underlying (non-succinct) NISC protocol and and the functionality g denotes the homomorphic evaluation $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$.

2. the ciphertexts $\mathsf{ct}_x, \mathsf{ct}_y$ are respectively encryptions of x, y using randomness r_x, r_y; and,
3. π is a valid proof of $\mathsf{ct}_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}_y)$ w.r.t. $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}})$ (as generated from r_{Del}).

If the checks pass, it decrypts $\mathsf{ct}_{\mathsf{out}}$ (by first generating $\mathsf{sk}_{\mathsf{FHE}}$ from r_{FHE}), obtaining the plaintext z, and finally outputs z. (If any of the checks fail, it instead outputs $\perp$.)

A summary of the message flow can be found in Figure 1.

A Subtlety in the Security Proof One subtle point that arises in the proof of security is that, to simulate a malicious sender, we need to simulate the ciphertext ct_x without knowledge of x. But the functionality of the underlying NISC takes as input the randomness used for both the key generation of $\mathsf{pk}_{\mathsf{FHE}}$ and for encrypting ct_x, and thus the functionality implicitly knows how to decrypt ct_x. A similar issue has arisen in the related context of constructing delegation schemes from FHE and related primitives (see [DLN+04]), where it was shown that so-called "spooky interactions" can arise, where a malicious sender (even though it does not how to decrypt the ciphertext) can in fact use this dependence to make the receiver output values that correlate in undesirable ways with the input x (in particular, in ways that would not have been possible if using an "idealized" FHE). Fortunately, in our context, we are able to overcome this issue by using the perfect correctness of the FHE scheme and soundness of our underlying delegation scheme to perform a carefully designed hybrid argument.

A bit more precisely, the key point is that when simulating a malicious sender in communication with an *honest* receiver, the receiver's public key and ciphertext ct_x will always be correctly generated (as such, we do not have to perform the checks involving the receiver to simulate the underlying NISC's output); furthermore, by soundness of delegation and the perfect correctness of the FHE, the decryption of $\mathsf{ct}_{\mathsf{out}}$ must equal $f(x, y)$ (with overwhelming probability) if π is accepting, so we can use this fact to show that decrypting $\mathsf{ct}_{\mathsf{out}}$ is actually *also* unnecessary. As such, we do not need to use either r_{FHE} or r_x to emulate the experiment for a malicious sender, and we can create (and prove security in) a hybrid functionality for the underlying NISC which is independent of this randomness (and only depends on $\mathsf{pk}_{\mathsf{FHE}}$).

2 Preliminaries

Notation. Let $\mathbb{N}$ denote the set of natural numbers (positive integers), and let $[n]$ denote the set of natural numbers at most n, or $\{1, 2, \ldots, n\}$. For $n \in \mathbb{N}$, we denote by 1^n the string of n ones, which will be used to provide a security parameter as input to an algorithm (this is by convention, so that the input length is bounded below by the security parameter). We assume the reader is familiar with polynomial-time and probabilistic polynomial time (PPT) algorithms. We say a function $\epsilon(\cdot)$ is *negligible* if, for any polynomial $p(\cdot)$, $\epsilon(n) < 1/p(n)$ for all sufficiently large $n \in \mathbb{N}$—that is, if $\epsilon(\cdot)$ is asymptotically smaller than any inverse polynomial.

2.1 Fully Homomorphic Encryption

Intuitively, a fully homomorphic encryption (FHE) scheme [Gen09] is an encryption scheme with the additional property that computations may, using the public key, be performed on ciphertexts for respective inputs in such a way that the result will be a ciphertext for the correct output. We formalize this as follows:

Definition 1 (based on [Gen09])**.** *A* ***fully homomorphic encryption*** *(FHE)* ***scheme*** *consists of a tuple of algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$*, where* Gen*,* Enc *are PPT and* Eval*,* Dec *are (deterministic) polynomial-time algorithms, such that:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n; \rho)$*: takes the security parameter n as input and outputs a public key* pk *and secret key* sk*.*
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m; \rho)$*: takes as input a public key* pk *and a message $m \in \{0, 1\}$, and outputs a ciphertext* ct*. (For multi-bit messages $\overrightarrow{m} \in \{0, 1\}^{p(n)}$, we let* $\overrightarrow{\mathsf{ct}} \leftarrow \mathsf{Enc}(\mathsf{pk}, \overrightarrow{m})$ *be such that* $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, m_i)$*.)*
- $\mathsf{ct}' = \mathsf{Eval}(\mathsf{pk}, C, \overrightarrow{\mathsf{ct}})$*: takes as input a list of ciphertexts* $\overrightarrow{\mathsf{ct}}$ *and a circuit description C of some function to evaluate and outputs a ciphertext* ct'*.*
- $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: takes as input a ciphertext* ct *and outputs a message m'.*

We furthermore require that the following properties are satisfied:

1. ***Full homomorphism:*** *There exist sets of boolean circuits $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$, negligible function $\epsilon(n)$, and polynomial $p(\cdot)$ such that $\mathcal{C} = \bigcup_n \mathcal{C}_n$ includes the set of all arithmetic circuits over* $\mathsf{GF}(2)$[7]*, and, for any $n \in \mathbb{N}$, we have that, for all $C \in \mathcal{C}_n$ and $\overrightarrow{m} \in \{0, 1\}^{p(n)}$:*

$$Pr[z \neq C(\overrightarrow{m}) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n), \overrightarrow{\mathsf{ct}} \leftarrow \mathsf{Enc}(\mathsf{pk}, \overrightarrow{m}),$$
$$z \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(C, \mathsf{pk}, \overrightarrow{\mathsf{ct}}))] < \epsilon(n),$$

   Furthermore, if this probability is identically zero, we refer to the scheme as having ***perfect correctness****.*

2. ***Compactness:*** *There exists a polynomial $q(\cdot)$ such that the output length of* Eval *given (any number of) inputs generated with security parameter n is at most $q(n)$.*

[7] $\mathsf{GF}(2)$ is the set of arithmetic circuits consisting only of $+$ and $\times$ gates over the field $\mathbb{F}_2$.

Definition 2 (based on [GM84]). *We say that an FHE* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is* ***secure*** *if, for all non-uniform PPT D, there exists a negligible $\epsilon(\cdot)$ such that for any $n \in \mathbb{N}$:*

$$|Pr[D(1^n, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 0)) = 1] - Pr[D(1^n, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, 1)) = 1]| < \epsilon(n)$$

over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$*. If this condition holds also with respect to subexponential-size distinguishers D (i.e., algorithms implemented by circuits of size poly(2^{n^ϵ}) for some $\epsilon > 0$), we refer to the scheme as being* ***subexponentially secure****.*

We have the following consequence for encryptions of $\mathsf{poly}(n)$-bit messages $\overrightarrow{m_0}, \overrightarrow{m_1}$:

Fact 1. *If an FHE scheme* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is secure (resp., subexponentially secure), then, for any polynomial $p(\cdot)$ and for any non-uniform PPT (resp., subexponential-size) $(\mathcal{A}, D)$ where $\mathcal{A}$ outputs messages $\overrightarrow{m_0}, \overrightarrow{m_1} \in \{0,1\}^{p(n)}$ for polynomial $p(\cdot)$, there exists a negligible $\epsilon(\cdot)$ such that for any $n \in \mathbb{N}$:*

$$|Pr[D(1^n, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \overrightarrow{m_0})) = 1] - Pr[D(1^n, \mathsf{pk}, \mathsf{Enc}(\mathsf{pk}, \overrightarrow{m_1})) = 1]| < \epsilon(n)$$

where

$$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n), (\overrightarrow{m_0}, \overrightarrow{m_1}) \leftarrow \mathcal{A}(1^n, \mathsf{pk})$$

We can construct an FHE scheme with all of the above properties based on the Learning With Errors (LWE) assumption:

Theorem 3 ([BGV12, GKP$^+$13, AEKP19]). *Based on computational (resp., subexponential) hardness of the Learning With Errors assumption, there exists a secure (resp., subexponentially secure) fully homomorphic encryption scheme satisfying perfect correctness.*

2.2 Adaptive Delegation Schemes

A delegation scheme allows for the effective "outsourcing" of computation from one party to another; that is, using delegation, the sender can compute both the correct result of some (possibly expensive) computation on a receiver's input and a (short) proof which can convince the receiver of the correctness of the computation without requiring the receiver to perform the computation themselves. We consider a notion of delegation with the additional property, formalized in [BHK17], that the functionality $f(\cdot)$ whose computation is to be delegated can be decided *adaptively* after the keys pk, sk are computed (i.e., the key-generation algorithm Gen is independent from f). Formally:

Definition 3 (based on [BHK17]). *An* ***adaptive delegation scheme*** *is given by a triple of algorithms* $(\mathsf{Gen}, \mathsf{Comp}, \mathsf{Ver})$*, where* Comp *and* Ver *are (deterministic) polynomial-time algorithms and* Gen *is PPT, such that:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n; \rho)$ *takes as input a security parameter n and probabilistically outputs a public key* pk *and secret key* sk*.*
- $(y, \pi, 1^T) \leftarrow \mathsf{Comp}(\mathsf{pk}, f, \overrightarrow{x})$ *takes as input a Turing machine description of the functionality f to be computed, as well as the inputs $\overrightarrow{x}$ to f, and produces a result y which the sender claims to be the result of the computation, a $\mathsf{poly}(n)$-size proof π of its correctness, and the running time T of the computation in unary.*

- $\{\mathsf{Accept}, \mathsf{Reject}\} \leftarrow \mathsf{Ver}(\mathsf{sk}, f, \overrightarrow{x}, y, \pi, T)$ *takes as input the functionality f to be computed, inputs $\overrightarrow{x}$, result y, proof π, and running time T, and returns* Accept *or* Reject *depending on whether π is a valid proof of $f(\overrightarrow{x}) = y$.*

Furthermore, we require the following properties:

1. ***Completeness:*** *There exists a negligible function $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$, any f computable by a Turing machine that runs in time at most 2^n, and any $\overrightarrow{x}$ in the domain of f:*

$$Pr\left[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n); (\pi, y, 1^T) = \mathsf{Comp}(\mathsf{pk}, f, \overrightarrow{x}) : \mathsf{Ver}(\mathsf{sk}, f, \overrightarrow{x}, \pi, y, T) = \mathsf{Reject}\right] < \epsilon(n)$$

   In addition, if the above probability is identically zero, we say that the adaptive delegation scheme satisfies ***perfect completeness****.*

2. ***Correctness:*** *For any $n \in \mathbb{N}$, any f computable by a Turing machine that runs in time at most 2^n, and any $\overrightarrow{x}$ in the domain of f:*

$$Pr[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n) : \mathsf{Comp}(\mathsf{pk}, f, \overrightarrow{x}) = (f(\overrightarrow{x}), \cdot, \cdot)] = 1$$

3. ***Soundness:*** *For any non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$:*

$$\begin{aligned}\Pr\big[(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n), (f, \overrightarrow{x}, y_1, y_2, \pi_1, \pi_2, 1^{T_1}, 1^{T_2}) \leftarrow \mathcal{A}(1^n, \mathsf{pk}) :& \\ T < 2^n \wedge \mathsf{Ver}(\mathsf{sk}, f, \overrightarrow{x}, y_1, \pi_1, T_1) = \mathsf{Accept}& \\ \wedge \mathsf{Ver}(\mathsf{sk}, f, \overrightarrow{x}, y_2, \pi_2, T_2) = \mathsf{Accept} \wedge y_1 \neq y_2\big] < \epsilon(n)&\end{aligned}$$

   Furthermore, if this condition holds with respect to subexponential-size adversaries, we say that the scheme is ***subexponentially sound****.*

A construction of an adaptive delegation scheme with perfect completeness can be found in the work of Brakerski et al. [BHK17], and is based on a secure private information retrieval (PIR) scheme, which in turn can be constructed based on a leveled FHE scheme (including the one presented in Theorem 3). Hence:

Theorem 4 ([BGV12, GKP+13, BHK17, AEKP19])**.** *Based on computational (resp., subexponential) hardness of the Learning With Errors assumption, there exists a sound (resp., subexponentially sound) adaptive delegation scheme satisfying perfect completeness.*

2.3 Non-Interactive Secure Computation

Definition 4 (based on [Yao82, GMW87, BGI+17])**.** *A* ***non-interactive two-party computation protocol*** *for computing some functionality $f(\cdot, \cdot)$ (where f is computable by a polynomial-time Turing machine) is given by three PPT algorithms $(\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ defining an interaction between a sender S and a receiver R, where only R will receive the final output. The protocol will have common input 1^n (the security parameter); the receiver R will have input x, and the sender will have input y. The algorithms $(\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ are such that:*

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

Furthermore, we require the following property:

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

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

Defining non-interactive *secure* computation will require us to add a security definition, which we formalize as follows:

Security. We adopt a standard notion of *simulation-based security*, with the relaxation that we allow superpolynomial-time simulation (as pioneered by [Pas03, PS04]). We define security by comparing two experiments conducted between the sender and receiver, either of whom may be corrupted and act arbitrarily (while the other is honest and follows the protocol). In the *real* experiment, the two parties will perform the actual protocol; in the *ideal* experiment, the two parties will instead send their inputs to a "trusted third party" who performs the computation and returns the result only to, in this case (because the protocol is one-sided), the receiver. Informally, we say that a protocol is secure if, for any adversary $\mathcal{A}$ against the *real* experiment, acting either as the sender or receiver, there is a simulated adversary $\mathcal{S}$ in the *ideal* experiment which produces a near-identical (i.e., computationally indistinguishable) result; intuitively, if this is the case, we can assert that the real adversary cannot "learn" anything more than they could by interacting with a trusted intermediary. Let us formalize this notion for the case of SNISC:

- Let the *real* experiment be defined as an interaction between a sender S with input y and a receiver R with input x, defined 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).

- Let the *ideal* experiment be defined as an interaction between a sender S, a receiver R, and a *trusted party* $\mathcal{T}_f$, defined 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.

Given the above, we can now formally define non-interactive secure computation:

Definition 5 (based on [Yao82, GMW87, Pas03, PS04, BGI+17])**.** *Given a function $T(\cdot)$, a non-interactive two-party protocol $\Pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ between a sender S and a receiver R, and functionality $f(\cdot, \cdot)$ computable by a polynomial-time Turing machine, we say that Π* ***securely computes f with $T(\cdot)$-time simulation****, or that Π is a* ***non-interactive secure computation*** *(NISC)* ***protocol (with $T(\cdot)$-time simulation)*** *for computing f, if Π is a non-interactive two-party computation protocol for computing f and, for any polynomial-time adversary $\mathcal{A}$ corrupting party $I \in \{S, R\}$, there exists a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}$ such that, for any $T(n) \cdot \mathsf{poly}(n)$-time algorithm $D : \{0, 1\}^* \to \{0, 1\}$, there exists negligible $\epsilon(\cdot)$ such that for any $n \in \mathbb{N}$ and any inputs $x, y \in \{0, 1\}^n, z \in \{0, 1\}^*$, we have:*

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

where the experiments and distributions Out are as defined above.

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

Succinctness. The defining feature of our construction will be a notion of *succinctness*; specifically, for functionality $f(\cdot, \cdot)$ with Turing machine description M and running time bounded by T_f, we show the existence of a NISC protocol $\Pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ for computing f whose message length (i.e., the combined output length of NISC_1 and NISC_2) and total receiver running time on input 1^n are relatively short and essentially independent of the running time of f. Formally:

Definition 6. *We say that a NISC protocol $\Pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ has* ***communication complexity*** *$\rho(\cdot)$ if, for any $n \in \mathbb{N}$, $x, y \in \{0, 1\}^n$, and $z \in \{0, 1\}^*$, the outputs of $\mathsf{NISC}_1(1^n, x)$ and $\mathsf{NISC}_2(1^n, y, z)$ contain at most $\rho(n)$ bits.*

We shall define a NISC protocol which, given functionality $f : \{0, 1\}^n \times \{0, 1\}^n \leftarrow \{0, 1\}^{\ell(n)}$ computable by a Turing machine M with running time $T_f(n)$, features communication complexity and receiver running time bounded above by $p(n, \log(T_f(n)), |M|, \ell(n))$ for an *a priori* fixed polynomial p.

There exist *non-succinct* non-interactive secure computation protocols in the standard model based on a notion of "weak oblivious transfer" ([BGI+17]), which in turn can be based on subexponential security of the Learning With Errors assumption [BD18]:

Theorem 5 ([BGI$^+$17, BD18], see also Appendix A)**.** *Assuming subexponential hardness of the Learning With Errors assumption, for any functionality $f(\cdot,\cdot)$ computable by a polynomial-time Turing machine there exists a (non-succinct) non-interactive secure computation protocol for computing f with quasi-polynomial simulation.*

We note that this theorem essentially follows from [BGI$^+$17, BD18]; however, [BGI$^+$17] required as an additional assumption the existence of an *onto* one-way function. In Appendix A, we present a variant which demonstrates how to prove Theorem 5 without this added assumption.

3 Protocol

We state our main theorem:

Theorem 6. *Assuming subexponential hardness of the Learning With Errors assumption, there exists polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that, for any polynomials $T_f(\cdot)$ and $\ell(\cdot)$ and any Turing machine M with running time bounded by $T_f(\cdot)$ computing functionality $f(\cdot,\cdot) : \{0,1\}^n \times \{0,1\}^n \leftarrow \{0,1\}^{\ell(n)}$, there exists a non-interactive secure computation protocol for computing f with quasi-polynomial simulation which is additionally* succinct *in that both its communication complexity and the running time of the honest receiver are at most $p(n, log(T_f(n)), |M|, \ell(n))$.*

We propose the protocol Π given in Figure 2 for secure non-interactive secure computation of a function $f(x,y)$ given a receiver input x and sender input y, where Π shall use the following primitives:

- Let $\pi = (\mathsf{NISC}_1, \mathsf{NISC}_2, \mathsf{NISC}_3)$ be a non-succinct NISC protocol with $T(n)$-time simulation for $T(n) = n^{\log^c(n)}$ (i.e., quasi-polynomial simulation), whose functionality h will be determined in the first round of the protocol. (The existence of such a primitive is guaranteed by Theorem 5 under subexponential LWE.)
- Let $(\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}})$ be a fully homomorphic encryption scheme satisfying perfect correctness, compactness, and subexponential security (in particular, with respect to $T(n) \cdot \mathsf{poly}(n)$-time adversaries). (The existence of such a primitive is guaranteed by Theorem 3 under subexponential LWE.)
- Let $(\mathsf{Gen}_{\mathsf{Del}}, \mathsf{Comp}_{\mathsf{Del}}, \mathsf{Ver}_{\mathsf{Del}})$ be an adaptive delegation scheme with perfect completeness, correctness, and subexponential soundness (in particular, with respect to $T(n) \cdot \mathsf{poly}(n)$-time adversaries). (The existence of such a primitive is guaranteed by Theorem 4 under subexponential LWE.)

4 Proof

Overview. After first proving the succinctness and correctness of the protocol, we turn to proving its security. We do this in two steps. In the first step, we consider a "hybrid" model in which the underlying NISC protocol is replaced by an "ideal" third party $\mathcal{T}_h$. If the underlying protocol were universally composable [Can01], this step would be trivial; unfortunately, it is not, so we need to

Input: The receiver R and the sender S are given input $x, y \in \{0,1\}^n$, respectively, and both parties have common input 1^n.
Output: R receives $f(x,y)$.

Round 1: R proceeds as follows:

1. Generate random coins $r_{\mathsf{FHE}} \leftarrow \{0,1\}^*$ and compute $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{sk}_{\mathsf{FHE}}) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$.
2. Let T_g denote the running time of the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$, and let $\lambda = \max(n, \log(T_g))$. Generate random coins $r_{\mathsf{Del}} \leftarrow \{0,1\}^*$ and compute $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$.
3. Generate random coins $r_{\mathsf{Enc}(x)} \leftarrow \{0,1\}^*$ and compute $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$.
4. Generate message $\mathsf{msg}_1 \leftarrow \mathsf{NISC}_1(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$ to compute the functionality h described in Figure 3.
5. Send $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$ to S.

Round 2: S proceeds as follows:

1. Generate random coins $r_{\mathsf{Enc}(y)} \leftarrow \{0,1\}^*$ and compute $\mathsf{ct}_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y; r_{\mathsf{Enc}(y)})$.
2. Compute $(\mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, 1^T) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$ for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$.
3. Generate message $\mathsf{msg}_2 \leftarrow \mathsf{NISC}_2(y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, r_{\mathsf{Enc}(y)}, T)$ to compute the functionality h described in Figure 3.
4. Send msg_2 to R.

Output phase: R proceeds as follows:

1. Compute $\mathsf{out} = \mathsf{NISC}_3(\mathsf{msg}_2)$ and return the result.

Figure 2: Protocol Π for succinct non-interactive secure computation.

Input: The receiver R has input $(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$, and the sender S has input $(y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, r_{\mathsf{Enc}(y)}, T)$
Output: Either a message out or the special symbol $\perp$.

Functionality:

1. Verify that all of the following checks hold. If any fail, return $\perp$.
   (a) $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$
   (b) $(\mathsf{pk}_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$
   (c) $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$
   (d) $\mathsf{ct}_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y; r_{\mathsf{Enc}(y)})$
2. Compute $(\cdot, \mathsf{sk}_{\mathsf{FHE}}) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$ and $(\cdot, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$.
3. If $\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, T) = \mathsf{Reject}$ for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$, then return $\perp$.
4. Compute $\mathsf{out} = \mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}})$ and return the result.

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

take care to formally reduce this transformation to the simulation-based security of the underlying protocol. Crucially, this will rely on the fact that we restrict our attention to two-round protocols.

Next, in the second step, we can create and prove the respective simulators for a corrupted sender and corrupted receiver in the $\mathcal{T}_h$-hybrid model. The corrupted receiver case follows in a fairly straightforward way, relying on the perfect correctness and completeness of the delegation and FHE schemes. The corrupted sender case, however, has some interesting subtleties in the reduction, and in fact will require another hybrid with a slightly different third party $\mathcal{T}_{h'}$ to complete; we discuss these subtleties in more detail when they arise during the proof.

We begin the formal proof by proving that the protocol Π is *succinct*:

Lemma 1. *There exists polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that, for any polynomials $T_f(\cdot)$ and $\ell(\cdot)$ and any Turing machine M with running time bounded by $T_f(\cdot)$ computing functionality $f(\cdot,\cdot) : \{0,1\}^n \times \{0,1\}^n \leftarrow \{0,1\}^{\ell(n)}$, the respective non-interactive secure computation protocol Π has communication complexity and honest receiver running time bounded above by $p(n, log(T_f(n)), |M|, \ell(n))$.*

Proof. To lead, we point out that, while $T_f(n)$ and $\ell(n)$ are given to be $\mathsf{poly}(n)$, we deliberately quantify them after $p(n)$ in order to treat them separately in the analysis below, as we specifically wish to show that the communication complexity of Π is at most polylogarithmic in the running time $T_f(n)$ for any possible polynomial-time Turing machine computable functionality $f(\cdot,\cdot)$. Furthermore, we assume without loss of generality that the input lengths provided to sub-algorithms are correct, as in the event that an adversary provides incorrectly sized inputs the algorithms may simply abort.

We begin by analyzing the communication complexity, as succinctness of the receiver's running time will follow immediately from this analysis. Aside from messages msg_1 and msg_2 for the underlying NISC π, the only communication consists of the public keys $\mathsf{pk}_{\mathsf{FHE}}$ and $\mathsf{pk}_{\mathsf{Del}}$ and the ciphertext ct_x. $\mathsf{pk}_{\mathsf{FHE}}$ has length $\mathsf{poly}(n)$ since $\mathsf{Gen}_{\mathsf{FHE}}$ is a polynomial-time algorithm running on input 1^n, and the ciphertext ct_x (which consists of a ciphertext for each bit in $x \in \{0,1\}^n$) also has length $\mathsf{poly}(n)$ since $\mathsf{Enc}_{\mathsf{FHE}}$ is polynomial-time and is run on inputs of length $\mathsf{poly}(n)$. $\mathsf{pk}_{\mathsf{Del}}$ will have length $\mathsf{poly}(n, \log(T_f))$; specifically, its length is given to be $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_g))$, where T_g is the running time of the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$ with inputs generated from common input 1^n. However, since $\mathsf{pk}_{\mathsf{FHE}}$ has $\mathsf{poly}(n)$ length, the input ciphertexts both have $\mathsf{poly}(n)$ length by the efficiency of $\mathsf{Enc}_{\mathsf{FHE}}$, and f in this case is given as a *circuit* description, which will have size $\mathsf{poly}(T_f(n))$, we have by the efficiency of $\mathsf{Eval}_{\mathsf{FHE}}$ that $T_g = \mathsf{poly}(n, T_f(n))$, implying $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_f(n)))$.

So it suffices now to bound the length of the NISC messages msg_1 and msg_2. Specifically, even for a *non-succinct* NISC protocol π, the honest sender and receiver must be efficient, and so the message length is still at most polynomial in the input length and running time of the functionality h. We argue that these are $\mathsf{poly}(n, \log(T_f(n)), |M|, \ell(n))$ to complete the proof of the claim:

- The input length to π is given as the size of the inputs $(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$ from the receiver and $(y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, r_{\mathsf{Enc}(y)}, T)$ from the sender. x and y have length n by assumption. $\mathsf{pk}_{\mathsf{FHE}}$, ct_x, and ct_y have length $\mathsf{poly}(n)$ as argued above, and $\mathsf{ct}_{\mathsf{out}}$ (which consists of a ciphertext output from $\mathsf{Eval}_{\mathsf{FHE}}$ for each bit of $f(x,y) \in \{0,1\}^{\ell(n)}$) has length $\mathsf{poly}(n, \ell(n))$ by the compactness of the underlying FHE scheme. $\mathsf{pk}_{\mathsf{Del}}$ has length $\mathsf{poly}(n, \log(T_f(n)))$ as argued above, and π_{Del} also has length $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_f(n)))$; T will have size $\lambda = \mathsf{poly}(n, \log(T_f(n)))$ as $T \le 2^\lambda$ is required by the properties of the delegation scheme. Lastly, the randomness $r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)}, r_{\mathsf{Enc}(y)}$ cannot have length greater than the running times of the respective algorithms $\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Gen}_{\mathsf{Del}}, \mathsf{Enc}_{\mathsf{FHE}}$, all of which we have already noted are at most $\mathsf{poly}(n, \log(T_f(n)))$.
- To bound the running time of the functionality h, notice that it consists of the following:
  - $\mathsf{Gen}_{\mathsf{FHE}}$ (run twice), $\mathsf{Enc}_{\mathsf{FHE}}$ (run $2n$ times, once for each bit of x and y), $\mathsf{Eval}_{\mathsf{FHE}}$ (run $\ell(n)$ times, once for each bit of out), all of which are efficient algorithms run on inputs of at most length $\mathsf{poly}(n)$ (and hence have running time $\mathsf{poly}(n)$);
  - $\mathsf{Dec}_{\mathsf{FHE}}$ (run $\ell(n)$ times), which has inputs $\mathsf{sk}_{\mathsf{FHE}}$ with size $\mathsf{poly}(n)$ and $\mathsf{ct}_{\mathsf{out}}$ with size $\mathsf{poly}(n, \ell(n))$, and hence has running time $\mathsf{poly}(n, \ell(n))$;
  - $\mathsf{Gen}_{\mathsf{Del}}$ (run twice), which runs in time $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_f(n)))$;
  - $\mathsf{Ver}_{\mathsf{Del}}$ (run once), which, given inputs $\mathsf{sk}_{\mathsf{Del}}, \pi_{\mathsf{Del}}$ of size $\mathsf{poly}(\lambda) = \mathsf{poly}(n, \log(T_f(n)))$, $\mathsf{ct}_x, \mathsf{ct}_y$ of size $\mathsf{poly}(n)$, $\mathsf{ct}_{\mathsf{out}}$ of size $\mathsf{poly}(n, \ell(n))$, g (the description of $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}($ $\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$, where we here interpret f as the Turing machine M) of size $\mathsf{poly}(|M|)$, and $T \le 2^\lambda$ of size at most $\lambda = \mathsf{poly}(n, \log(T_f(n)))$, has running time which is at most $\mathsf{poly}(n, \log(T_f(n)), |M|, \ell(n))$;

  and a $\mathsf{poly}(n)$ number of comparisons between input values and function outputs which have already been established to have at most $\mathsf{poly}(n, \log(T_f(n)))$ length.

The above shows that the communication complexity of Π is succinct. Furthermore, as the honest receiver runs only $\mathsf{Gen}_{\mathsf{FHE}}$, $\mathsf{Gen}_{\mathsf{Del}}$, $\mathsf{Enc}_{\mathsf{FHE}}$, and the (efficient) receiver protocol for the underlying NISC on the aforementioned inputs, and as we have already established that all of these algorithms have running time $\mathsf{poly}(n, \log(T_f(n)), |M|, \ell(n))$, the receiver will inherit the same running time bound. $\square$

Towards proving security for Π, let $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n,x,y,z)$ denote the random variable, taken over all randomness used by the honest party and the adversary, of the outputs of the honest receiver (if $I=S$) and the adversary in the execution of protocol Π given adversary $\mathcal{A}$ controlling corrupted party $I \in \{S,R\}$, receiver input x, sender input y, and adversary auxiliary input z. Let $\mathsf{Exec}_{\Pi,\mathcal{A},I}(1^n,x,y,z)$ denote the respective experiment.

Let us also define the "ideal" execution by letting $\mathcal{T}_f$ denote the ideal functionality corresponding to the computation target $f(x,y)$ and letting Π_f be the "ideal" version of the protocol where R sends x to $\mathcal{T}_f$, S sends y to $\mathcal{T}_f$, and then R finally outputs the result out output by $\mathcal{T}_f$. We want to show the following theorem:

Theorem 7. *Assume, given functionality $f(\cdot,\cdot)$, the respective protocol Π described in Figure 2 and the assumptions required in Theorem 6, and let $T(\cdot)$ be such that the underlying NISC π is secure with $T(\cdot)$-time simulation. For any efficient 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 non-uniform polynomial-time distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$, $x,y \in \{0,1\}^n$, and auxiliary input z, D distinguishes the distributions $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n,x,y,z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n,x,y,z)$ with at most probability $\epsilon(n)$.*

Notice that correctness of Π holds trivially from the perfect correctness of the underlying FHE, the correctness and perfect completeness of the underlying adaptive delegation scheme, and the correctness of the underlying NISC protocol π; hence, Theorem 7, which proves security, and Lemma 1, which proves succinctness, will in conjunction directly imply Theorem 6 (where quasi-polynomial simulation results from our use of an underlying NISC protocol with quasi-polynomial simulation, as given in Theorem 5). The remainder of the section, then, is devoted to proving Theorem 7.

4.1 Comparing Real and Hybrid Executions

We begin by defining a "trusted third party" $\mathcal{T}_h$ which executes the ideal functionality for h—that is, given the corresponding sender and receiver inputs, $\mathcal{T}_h$ outputs the correct value of h computed on those inputs. Our first task is to show, then, that the "real" experiment's outputs $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n,x,y,z)$ cannot be distinguished from those of a "hybrid" experiment, which we shall denote by $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n,x,y,z)$.

Formally, we let Π_h denote a protocol which is identical to Π with the exception that, in rounds 1 and 2, rather than generating msg_1 and msg_2, R and S instead send the respective inputs to $\mathcal{T}_h$, and, in the output phase, R receives and returns the output from $\mathcal{T}_h$ rather than unpacking msg_2. We then prove the following lemma:

Lemma 2. *For any efficient adversary $\mathcal{A}$ corrupting party $I \in \{S,R\}$, there exists a $T(n)\cdot\mathsf{poly}(n)$-time adversary $\mathcal{A}'$ such that, for any non-uniform polynomial-time distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$, $x,y \in \{0,1\}^n$, and auxiliary input z, D distinguishes the distributions $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n,x,y,z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n,x,y,z)$ with at most probability $\epsilon(n)$.*

Proof. We separate into two cases, based on whether $I=R$ (the receiver is corrupted) or $I=S$ (the sender is corrupted).

Corrupted Receiver. In this case we begin by, given some adversary $\mathcal{A}$ against the real experiment $\mathsf{Exec}_{\Pi,\mathcal{A},R}(1^n, x, y, z)$, defining an adversary $\mathcal{A}_h$ against the underlying 2PC protocol π. Without loss of generality, let $\mathcal{A}$ be a deterministic algorithm which uses the auxiliary input z as its source of randomness for r_{FHE}, r_{Del}, and $r_{\mathsf{Enc}(x)}$. $\mathcal{A}_h(1^n, z)$ will behave identically to $\mathcal{A}(1^n, z)$ (acting as the corrupted receiver), with the exception that $\mathcal{A}_h$ will only send msg_1 in round 1. On receiving msg_2 from the honest sender, $\mathcal{A}_h$ will run the output phase of $\mathcal{A}$ once again, using msg_2 as the input from the second round, to determine $\mathcal{A}$'s final output $\mathsf{out}_{\mathcal{A}}$ and return the result.

Now, consider the following adversary $\mathcal{A}'_{\mathsf{Real}}(1^n, z)$ in the real experiment, which runs $\mathcal{A}_h$:

1. Run $\mathcal{A}_h(1^n, z)$, which will start by producing a message msg_1 for π. Also run $\mathcal{A}(1^n, z)$ to produce a message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \cdot)$, and send $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$ to the sender S.

2. S will return a message msg_2 for π. Run the output phase of $\mathcal{A}_h$ on this message.

3. $\mathcal{A}_h$ will output $\mathsf{out}_{\mathcal{A}_h}$; return it.

Also consider a $T(n) \cdot \mathsf{poly}(n)$-time adversary $\mathcal{A}'$ in the hybrid experiment $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$ which runs the ($T(n) \cdot \mathsf{poly}(n)$-time) simulator $\mathcal{S}_h$ corresponding to the adversary $\mathcal{A}_h$ (as guaranteed by the definition of simulation-based security for π). $\mathcal{A}'(1^n, z)$ will do as follows:

1. Run $\mathcal{S}_h(1^n, z)$, which will start by producing a message m_1 to send to the ideal functionality $\mathcal{T}_h$; forward this message. Also run $\mathcal{A}(1^n, z)$ to produce a message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$, and send $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ to the sender S.

2. S will provide its input to $\mathcal{T}_h$, and subsequently $\mathcal{T}_h$ will return a result out. Forward out to $\mathcal{S}_h$.

3. $\mathcal{S}_h$ will return a simulated message $\mathsf{out}_{\mathcal{S}}$; return it.

We can use these adversaries to show that $\mathsf{Out}_{\Pi,\mathcal{A},R}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$ cannot be distinguished with non-negligible probability, based on the following two claims:

Claim 1. $\mathsf{Out}_{\Pi,\mathcal{A},R}(1^n, x, y, z)$ *and* $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)$ *are identically distributed.*

Proof. Importantly, recall that the adversaries $\mathcal{A}$ and $\mathcal{A}_h$ are deterministic and use z as their source of randomness. We begin by reproducing the experiment $\mathsf{Exec}_{\Pi,\mathcal{A},R}(1^n, x, y, z)$ for clarity:

1. Run $\mathcal{A}(1^n, z)$ to produce a message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$, and send it to the sender S.

2. S will return a message msg_2 for π. Run the output phase of $\mathcal{A}$ on this message.

3. $\mathcal{A}$ will output $\mathsf{out}_{\mathcal{A}}$; return it.

There are two semantic differences between the two experiments. Namely, $\mathcal{A}'_{\mathsf{Real}}$ uses $\mathcal{A}_h(1^n, z)$ rather than $\mathcal{A}(1^n, z)$ to produce msg_1, as well as to produce the final output. However, by the definition of $\mathcal{A}_h$, for any auxiliary input z it is clearly the case that $\mathcal{A}_h(1^n, z)$ and $\mathcal{A}(1^n, z)$ compute msg_1 and their final output in an identical way; hence, the full experiments are completely identical. □

So it is equivalent to compare $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$. The following claim, then, yields the desired conclusion:

Claim 2. *For any polynomial-time non-uniform distinguisher D, there exists negligible $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$ and inputs x, y, z, the distributions $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$ cannot be distinguished by D with probability greater than $\epsilon(n)$.*

Proof. This will intuitively follow from the simulation-based security of the underlying protocol π.

Formally, assume for contradiction that there exists a non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there are inputs x, y, z such that D is able to distinguish the two distributions $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$ with probability $1/p(n)$. In this case, for each such n, there must exist some assignment r^* of the (honest) sender's randomness $r_{\mathsf{Enc}(y)}$ such that D distinguishes $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)|_{r^*}$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)|_{r^*}$ (which denote the respective experiments with $r_{\mathsf{Enc}(y)}$ fixed to r^*) with probability at least $1/p(n)$.

Given fixed x, y, z and fixed $r_{\mathsf{Enc}(y)} = r^*$, recall the inputs $x' = (x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$ and $y' = (y, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, r_{\mathsf{Enc}(y)}, T)$ provided by the receiver and sender (respectively) to the protocol π; all randomness used to generate these inputs is now fixed (since $r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)}$ are given by z), which means that each assignment of inputs x, y, z, r^* for Π corresponds uniquely to fixed inputs x', y', z for the underlying protocol π.

This implies that we can use the distinguisher D directly to break the simulation-based security of the underlying protocol π. Specifically, given the previously fixed x, y, z, r^*, consider the distributions $\mathsf{Out}_{\pi,\mathcal{A}_h,R}(1^n, x', y', z)$ and $\mathsf{Out}_{\pi_h,\mathcal{S}_h,R}(1^n, x', y', z)$, where π_h is the idealized version of π (as executed in Π_h) where both parties send their respective inputs x', y' to the functionality $\mathcal{T}_h$, which computes $h(x', y')$.

Since, by definition, the outputs of $\mathcal{A}'_{\mathsf{Real}}$ and $\mathcal{A}'$ (fixing inputs x, y, z, r^*) are given by the outputs of $\mathcal{A}_h$ and $\mathcal{S}_h$ (fixing the corresponding inputs x', y', z) in the respective experiments π and π_h, we have that

$$\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)|_{r^*} = \mathsf{Out}_{\pi,\mathcal{A}_h,R}(1^n, x', y', z)$$

and

$$\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)|_{r^*} = \mathsf{Out}_{\pi_h,\mathcal{S}_h,R}(1^n, x', y', z)$$

But, since D distinguishes $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},R}(1^n, x, y, z)|_{r^*}$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)|_{r^*}$ with probability $1/p(n)$, it must also distinguish $\mathsf{Out}_{\pi,\mathcal{A}_h,R}(1^n, x', y', z)$ and $\mathsf{Out}_{\pi_h,\mathcal{S}_h,R}(1^n, x', y', z)$ with the same probability. And, as this holds for infinitely many $n \in \mathbb{N}$, it follows that D contradicts the simulation-based security of the underlying protocol π (w.r.t. non-uniform polynomial-time distinguishers), which is a contradiction. $\square$

Corrupted Sender. Given an adversary $\mathcal{A}$ against the real experiment $\mathsf{Exec}_{\Pi,\mathcal{A},S}(1^n, x, y, z)$, let $\mathcal{A}_h$ as before be the respective adversary against simulation-based security of the 2PC protocol π. Now $\mathcal{A}_h$ receives message msg_1 from the honest receiver and must generate msg_2 using $\mathcal{A}$; however, $\mathcal{A}$ also requires the public parameters $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ from the receiver's first-round message as input. As such, we concatenate these parameters with the auxiliary input z provided to $\mathcal{A}_h$. Formally, we let $\mathcal{A}_h(1^n, (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z))$ given message msg_1 run $\mathcal{A}(1^n, z)$ given message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$. Then let $\mathcal{A}'_{\mathsf{Real}}(1^n, z)$ in the real experiment proceed as follows:

1. Receive a message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$ from the receiver R.
2. Run $\mathcal{A}_h(1^n, z' = (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z))$ with msg_1 as the input from the receiver; forward the message msg_2 from $\mathcal{A}_h$ to the sender.

3. Output whatever $\mathcal{A}_h$ outputs.

By simulation-based security, then, there is also a simulator $\mathcal{S}_h$ in the idealized experiment for π which sends the relevant inputs to the ideal functionality $\mathcal{T}_h$ but returns no output to the receiver. Let $\mathcal{A}'(1^n, z)$ be a $T(n) \cdot \mathsf{poly}(n)$-time adversary in the hybrid experiment $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ which does the following:

1. Receive a message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ from the receiver R.

2. Run $\mathcal{S}_h(1^n, z' = (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z))$. $\mathcal{S}_h$ will produce a message m_2 to send to the ideal functionality $\mathcal{T}_h$; forward it to $\mathcal{T}_h$.

3. Output whatever $\mathcal{S}_h$ outputs.

We must show that, given this $\mathcal{A}'$, both (1) the outputs of the adversary and (2) the outputs of the honest receiver R between $\mathsf{Exec}_{\Pi,\mathcal{A},S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ cannot be distinguished. The following claims imply the desired conclusion:

Claim 3. *$\mathsf{Out}_{\Pi,\mathcal{A},S}(1^n, x, y, z)$ and $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)$ are identically distributed.*

Proof. This follows directly from the fact that the adversaries $\mathcal{A}$ and $\mathcal{A}_h$ are deterministic and use z as their source of randomness, and the fact that $\mathcal{A}_h(1^n, (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z))$ given message msg_1 behaves identically to $\mathcal{A}(1^n, z)$ given message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, \mathsf{msg}_1)$, implying that msg_2 and the output of $\mathcal{A}$ (resp. $\mathcal{A}_h$) are identically distributed between the experiments. ☐

So it suffices to compare $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$, which we do as follows:

Claim 4. *For any polynomial-time non-uniform distinguisher D, there exists negligible $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$ and inputs x, y, z, the distributions $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ cannot be distinguished by D with probability greater than $\epsilon(n)$.*

Proof. Once again this will intuitively follow from the simulation-based security of the underlying protocol π.

Formally, assume for contradiction that there exists a non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there are inputs x, y, z such that D is able to distinguish the two distributions $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ with probability $1/p(n)$. As with the corrupted receiver case, for each such n, there must exist some assignment r^* of the (honest) receiver's randomness $r_R = (r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)})$ such that D distinguishes $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)|_{r^*}$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)|_{r^*}$ (which once again denote the respective experiments with r_S fixed to r^*) with probability at least $1/p(n)$.

Given x, y, z, and $r_R = r^*$ fixed, we must argue as before that this corresponds uniquely to fixed inputs x', y', z' for the underlying protocol π. In this case, this will follow from the fact that $\mathcal{A}'_{\mathsf{Real}}$ and $\mathcal{A}'$ run $\mathcal{A}_h$ and $\mathcal{S}_h$ (respectively) on the input $(1^n, z')$, where $z' = (\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x, z)$ is fully determined by z, x, and the randomness r_R consumed by the receiver in the first round (notice that this holds because $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ is part of the honest receiver's first message in the protocol Π). Furthermore, x' is uniquely determined by x and r_R, and y', the sender's input, is uniquely determined by z', y, and the sender's randomness (which, since the sender is malicious, is given by z).

As before, consider the distributions $\mathsf{Out}_{\pi,\mathcal{A}_h,R}(1^n, x', y', z')$ and $\mathsf{Out}_{\pi_h,\mathcal{S}_h,R}(1^n, x', y', z')$, where π_h is the idealized version of π (as executed in Π_h) where both parties send their respective inputs x', y' to the functionality $\mathcal{T}_h$. As before, the outputs of $\mathcal{A}'_{\mathsf{Real}}$ and $\mathcal{A}'$ (fixing inputs x, y, z, r^*) are given by the outputs of $\mathcal{A}_h$ and $\mathcal{S}_h$ (fixing the corresponding inputs x', y', z') in the respective experiments π and π_h; furthermore, by the definition of the protocols Π and Pi_h, the (honest) receiver's outputs are given by the outputs of π and π_h (respectively). So we once again determine that

$$\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)|_{r^*} = \mathsf{Out}_{\pi,\mathcal{A}_h,S}(1^n, x', y', z')$$

and

$$\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)|_{r^*} = \mathsf{Out}_{\pi_h,\mathcal{S}_h,S}(1^n, x', y', z')$$

which, since D distinguishes $\mathsf{Out}_{\Pi,\mathcal{A}'_{\mathsf{Real}},S}(1^n, x, y, z)|_{r^*}$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)|_{r^*}$ with probability $1/p(n)$, implies that it must also distinguish $\mathsf{Out}_{\pi,\mathcal{A}_h,S}(1^n, x', y', z')$ and $\mathsf{Out}_{\pi_h,\mathcal{S}_h,S}(1^n, x', y', z')$ with the same probability. Thus, since the above implies that there exist x', y', z' for infinitely many $n \in \mathbb{N}$ such that D distinguishes the respective distributions, it follows that D contradicts the simulation-based security of π (w.r.t. non-uniform polynomial-time distinguishers), a contradiction.

□

□

4.2 Comparing Hybrid and Ideal Executions

Next, we need to compare the hybrid execution $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$ to the "ideal" execution $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$ to finish the proof of Theorem 7.

Lemma 3. *For any $T(n) \cdot \mathsf{poly}(n)$-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 non-uniform polynomial-time distinguisher D, there exists a negligible function $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$, $x, y \in \{0,1\}^n$, and auxiliary input z, D distinguishes the distributions $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$ with at most probability $\epsilon(n)$.*

Proof. We again separate into two cases, based on whether $I = R$ (the receiver is corrupted) or $I = S$ (the sender is corrupted).

Corrupted Receiver. In this case, define a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}_R$ which does as follows:

1. Run the corrupted receiver $\mathcal{A}'$. $\mathcal{A}'$, in the first round, will output a message $(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}})$ to be sent to $\mathcal{T}_h$. Send x to the ideal functionality $\mathcal{T}_f$.

2. Receive an output message out from the ideal functionality $\mathcal{T}_f$. If out is $\perp$, return $\perp$ to $\mathcal{A}'$ (as the output of $\mathcal{T}_h$).

3. Verify the following. If any checks fail, return $\perp$ to $\mathcal{A}'$.

   (a) $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$
   (b) $(\mathsf{pk}_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$

(c) $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$

4. If all checks in the previous step pass, return out to $\mathcal{A}'$. Finally, output whatever $\mathcal{A}'$ outputs.

It suffices here to argue that the output which $\mathcal{S}_R$ returns to $\mathcal{A}'$ in the ideal experiment is identically distributed to the output which $\mathcal{T}_h$ would return to $\mathcal{A}'$ in the hybrid experiment, as this, combined with the observation that the only input $\mathcal{A}'$ receives (aside from the auxiliary input z) is the output from $\mathcal{T}_h$, allows us to conclude that $\mathcal{A}'$'s views in $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_R,R}(1^n, x, y, z)$ (and hence $\mathcal{A}'$'s outputs) are likewise identically distributed. We can argue this using the following claims:

Claim 5. *If S is honest, then, given messages $(x, r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}})$ and $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ from $\mathcal{A}'$, step (4) of $\mathcal{S}_R$ succeeds (i.e., does not return $\perp$) in Π_f if and only if all checks in step (1) of the functionality h described in Figure 3 succeed in the respective instance of Π_h.*

Proof. The "if" direction is trivial since the checks in step (4) of $\mathcal{S}_R$ are a strict subset of the checks in step (1) of h.

The "only if" direction follows from the assumption that S is honest, and will hence compute $\mathsf{ct}_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y; r_{\mathsf{Enc}(y)})$ correctly using the correct inputs. □

Claim 6. *If S is honest and all checks in step (1) of the functionality h described in Figure 3 succeed in Π_h, then, with probability 1, step (3) of the functionality h will not return $\perp$.*

Proof. Since step (1) is successful, we know that $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda, r_{\mathsf{Del}})$; moreover, since S is honest, we know that it must have computed $(\mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, 1^T) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$ correctly (and using the correct $\mathsf{pk}_{\mathsf{Del}}$ and ct_x, since the checks in step (1) passed). It follows by perfect completeness of the delegation scheme $(\mathsf{Gen}_{\mathsf{Del}}, \mathsf{Comp}_{\mathsf{Del}}, \mathsf{Ver}_{\mathsf{Del}})$ that

$$\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, T) = \mathsf{Accept}$$

as desired. □

Claim 7. *If S is honest and, in Π_h, all checks in step (1) of the functionality h described in Figure 3 succeed, and step (3) of the functionality h does not return $\perp$, then the value of out returned by step (4) of h will be equal to $f(x, y)$ with probability 1.*

Proof. Since S is honest and step (1) is successful, we know, as in the previous claim, that $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda, r_{\mathsf{Del}})$ and $(\mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, 1^T) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$. It follows by correctness of the delegation scheme $(\mathsf{Gen}_{\mathsf{Del}}, \mathsf{Comp}_{\mathsf{Del}}, \mathsf{Ver}_{\mathsf{Del}})$ that

$$\mathsf{ct}_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}_y) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, \mathsf{ct}_x, \mathsf{ct}_y)$$

It suffices to show that this will decrypt to the correct output $\mathsf{out} = f(x, y)$. This holds due to perfect correctness of the FHE scheme $(\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}})$; specifically, since ct_x and ct_y are encryptions of x and y, respectively:

$$\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}}) = \mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, \mathsf{ct}_x, \mathsf{ct}_y)) = f(x, y)$$

□

Chaining together Claims 5, 6, and 7 leads us to the conclusion that (by Claim 5), $\mathcal{S}_R$ returns $\perp$ in $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_R,R}(1^n, x, y, z)$ if and only if $\mathcal{T}_h$ would return $\perp$ (from step (1)) in the respective execution of $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$, and furthermore, if this event does not occur, then (by Claims 6 and 7 as well as the definition of $\mathcal{S}_R$) both $\mathcal{S}_R$ (in $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_R,R}(1^n, x, y, z)$) and $\mathcal{T}_h$ (in the respective execution of $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',R}(1^n, x, y, z)$) will return an output out that is precisely equal to $f(x, y)$, where x is the value sent by the adversary to $\mathcal{T}_h$ and y is the (honest) sender's input. This completes the argument for the case $I = R$.

Corrupted Sender. In the case $I = S$, define a $T(n) \cdot \mathsf{poly}(n)$-time simulator $\mathcal{S}_S$ which does as follows:

1. Generate $r_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}(x)} \leftarrow \{0, 1\}^*$
   and $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}}), (\mathsf{pk}_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}}), \mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, 0; r_{\mathsf{Enc}(x)})$
2. Run the corrupted sender $\mathcal{A}'$ using input $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$. $\mathcal{A}'$ will generate a message $(y', \mathsf{pk}'_{\mathsf{FHE}}, \mathsf{pk}'_{\mathsf{Del}}, \mathsf{ct}'_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, r'_{\mathsf{Enc}(y)}, T')$ to send to $\mathcal{T}_h$. Perform the following checks to verify this message, and return $\perp$ to $\mathcal{T}_f$ (causing it to output $\perp$) if any of them fail.
   (a) $\mathsf{pk}_{\mathsf{FHE}} = \mathsf{pk}'_{\mathsf{FHE}}$, $\mathsf{pk}_{\mathsf{Del}} = \mathsf{pk}'_{\mathsf{Del}}$, $\mathsf{ct}_x = \mathsf{ct}'_x$.
   (b) $\mathsf{ct}'_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y'; r'_{\mathsf{Enc}(y)})$
   (c) $\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, T') = \mathsf{Accept}$
   for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$.
3. Otherwise (if the above checks pass), send y' to $\mathcal{T}_f$. Finally, output whatever $\mathcal{A}'$ outputs.

As this case has interesting subtleties, we lead the formal proof with a brief overview. Recall that, for this case, we need not only to verify that the adversary $\mathcal{A}'$'s views in the experiments $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n, x, y, z)$ (and hence $\mathcal{A}'$'s outputs) cannot be distinguished, but also that the honest receiver R's outputs cannot be distinguished between the two experiments.

The natural way to do this would be to begin by creating a hybrid protocol Π'_h where the receiver, instead of sending a ciphertext of their input x in the first round, sends the corresponding ciphertext of 0 (as the simulator does when running $\mathcal{A}'$ in Π_f). Ostensibly, this would allow us to show that the output distributions between Π_h and Π'_h are close by using the CPA-security of the underlying FHE protocol to assert that the ciphertexts, and hence the views of $\mathcal{A}'$, are indistinguishable between the two experiments. And while this does indeed directly imply that the *adversary's* outputs are close, we run into an issue the moment we consider the *receiver's* output; specifically, the receiver's output is the output from the ideal functionality $\mathcal{T}_h$, which among other things depends on *the secret key* $\mathsf{sk}_{\mathsf{FHE}}$ *and the randomness* r_{FHE} *used to generate it.* In fact, this makes a reduction from Π'_h to the security of the FHE scheme impossible (using current techniques), since a hypothetical adversary simulating this functionality would only know $\mathsf{pk}_{\mathsf{FHE}}$.

Instead we will have to consider an alternate functionality h' which only depends on the public key $\mathsf{pk}_{\mathsf{FHE}}$ and does not use the randomness or secret key. Specifically, rather than decrypting the final result $\mathsf{ct}_{\mathsf{out}}$, h' will instead simply return $f(x, y')$. We then show that the output distribution of $\Pi_{h'}$ is *statistically close* to that of Π_h. Specifically, they are identical except when the adversary $\mathcal{A}'$ can force the ideal functionality h' to verify a proof π_{Del} of an incorrect ciphertext $\mathsf{ct}_{\mathsf{Out}}$—this implies that their statistical distance must be at most the (negligible) soundness error of delegation.[8] Now, given $\Pi_{h'}$, we can finally consider a protocol $\Pi'_{h'}$ where the receiver uses a ciphertext of 0; now that h' no longer depends on $\mathsf{sk}_{\mathsf{FHE}}$, the reduction to the CPA-security will go through (for both the adversary's and receiver's outputs), and we can lastly compare $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n, x, y, z)$ to show that, actually, the output distributions are identically distributed.

We continue to the formal proof. Let h' be the functionality defined as h, but with four key differences:

- h', instead of taking input r_{FHE} from the receiver, takes input $\mathsf{pk}_{\mathsf{FHE}}$.
- In step (1), instead of verifying that $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n, r_{\mathsf{FHE}})$, h' verifies that the sender's and receiver's inputs $\mathsf{pk}_{\mathsf{FHE}}$ match.
- In step (2), h' no longer computes $(\cdot, \mathsf{sk}_{\mathsf{FHE}}) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}})$.
- In step (4), h' returns $f(x, y)$ rather than $\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}})$.

Let $\Pi_{h'}$ be defined identically to Π_h except that both parties use the ideal functionality $\mathcal{T}_{h'}$ in place of $\mathcal{T}_h$ and the receiver inputs $\mathsf{pk}_{\mathsf{FHE}}$ to $\mathcal{T}_{h'}$ instead of r_{FHE} as specified above. We state the following claim:

Claim 8. *There exists negligible $\epsilon(\cdot)$ such that, for all $n \in \mathbb{N}$ and inputs x, y, z, the output distributions $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ are $\epsilon(n)$-statistically close.*

Proof. Intuitively, this will follow from the soundness of the delegation scheme $(\mathsf{Gen}_{\mathsf{Del}}, \mathsf{Comp}_{\mathsf{Del}}, \mathsf{Ver}_{\mathsf{Del}})$.

First, observe that the adversary's views in $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$, and thus the adversary's outputs, are identically distributed; hence, it suffices to argue about the honest receiver's output, i.e., the output of $\mathcal{T}_h$ or $\mathcal{T}_{h'}$.

Second, since the receiver R is honest, the fact that h' verifies that the sender's and receiver's inputs $\mathsf{pk}_{\mathsf{FHE}}$ match is equivalent to the verification in h of the sender's $\mathsf{pk}_{\mathsf{FHE}}$ (that $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n, r_{\mathsf{FHE}})$), since the receiver's input $\mathsf{pk}_{\mathsf{FHE}}$ will always be equal to $\mathsf{Gen}_{\mathsf{FHE}}(1^n, r_{\mathsf{FHE}})$. So the only change that can possibly affect the output of $\mathcal{T}_{h'}$ compared to $\mathcal{T}_h$ in the corrupted sender case is the fact that h' returns $f(x, y)$ rather than $\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}})$.

Now, assume for the sake of contradiction that there is some polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there exist x, y, z so that the ideal functionality's output is different between $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ with probability $1/p(n)$. We shall use this to construct a $T(n) \cdot \mathsf{poly}(n)$-time adversary $\mathcal{A}_{\mathsf{Del}}$ to break the soundness of the delegation scheme with probability $1/p(n)$.

Specifically, let $\mathcal{A}_{\mathsf{Del}}$ do as follows on input $(1^n, \mathsf{pk}_{\mathsf{Del}})$:

1. Generate $r_{\mathsf{FHE}}, r_{\mathsf{Enc}(x)} \leftarrow \{0,1\}^*$
   and $(\mathsf{pk}_{\mathsf{FHE}}, \cdot) = \mathsf{Gen}_{\mathsf{FHE}}(1^n; r_{\mathsf{FHE}}), \mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$.

[8] An attentive reader might wonder at this point why, in doing this, we are not simply backing ourselves into the same corner, since indeed $\mathcal{T}_h$ and even $\mathcal{T}_{h'}$ are very much dependent on the randomness r_{Del} and secret key $\mathsf{sk}_{\mathsf{Del}}$. The intuitive answer is that, unlike with the reduction to FHE, we are able to "outsource" the dependence on $\mathsf{sk}_{\mathsf{Del}}$ in $\mathcal{T}_{h'}$ to the security game for the soundness of delegation, allowing us to effectively *emulate* h' without said secret key in the adversary we construct.

2. Run the corrupted sender $\mathcal{A}'$ with sender input y, auxiliary input z, and first-round message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$. $\mathcal{A}'$ will generate a message $(y', \mathsf{pk}'_{\mathsf{FHE}}, \mathsf{pk}'_{\mathsf{Del}}, \mathsf{ct}'_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, r'_{\mathsf{Enc}(y)}, T')$ to send to the ideal functionality ($\mathcal{T}_h$ or $\mathcal{T}_{h'}$).

3. Run $(\mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, 1^T) \leftarrow \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y)$
for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$

4. Verify the following and abort if any are false.

   (a) $\mathsf{pk}_{\mathsf{FHE}} = \mathsf{pk}'_{\mathsf{FHE}}$, $\mathsf{pk}_{\mathsf{Del}} = \mathsf{pk}'_{\mathsf{Del}}$, $\mathsf{ct}_x = \mathsf{ct}'_x$
   (b) $\mathsf{ct}'_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y'; r'_{\mathsf{Enc}(y)})$

5. Otherwise, return $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$.

We claim that $\mathcal{A}_{\mathsf{Del}}$ returns a tuple $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$ such that $\mathsf{ct}_{\mathsf{out}} \neq \mathsf{ct}'_{\mathsf{out}}$ but $\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \pi_{\mathsf{Del}}, T) = \mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, T') = \mathsf{Accept}$—that is, $\mathcal{A}_{\mathsf{Del}}$ breaks soundness of the delegation scheme—precisely when h decrypts a ciphertext that is not equal to $\mathsf{ct}_{\mathsf{out}}$ as returned by $\mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}_y)$ for the corresponding functionality and inputs; furthermore, we claim that this is the only case where h and h' may not be identically distributed.

To verify this, we start by observing that the input to $\mathcal{A}'$ in step (2) of $\mathcal{A}_{\mathsf{Del}}$ is identically distributed to the inputs in the experiments $\mathsf{Exec}^{\mathcal{T}_h}_{\Pi_h, \mathcal{A}', S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'}, \mathcal{A}', S}(1^n, x, y, z)$, since $\mathsf{pk}_{\mathsf{Del}}$ is honestly generated and the receiver is honest. Furthermore, given the message from $\mathcal{A}'$ to the ideal functionality, as well as the fact that R is honest, we can assert that the checks in step (4) of $\mathcal{A}_{\mathsf{Del}}$ are equivalent to the checks in step (1) of h or h', since the receiver's inputs $\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x$ are guaranteed to be honestly generated. So, comparing $\mathcal{T}_h$ and $\mathcal{T}_{h'}$ for a particular interaction, there are four possible outcomes, which we shall analyze:

1. Step (1) of h or h' fails, in which case both return $\perp$ (and $\mathcal{A}_{\mathsf{Del}}$ will abort).

2. Step (1) succeeds, but the verification in step (3) fails, in which case both will return $\perp$ (and $\mathcal{A}_{\mathsf{Del}}$ will produce output $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$ which is *rejected* because $(\mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}})$ fails to verify).

3. Steps (1) and (3) succeed, and $\mathsf{ct}'_{\mathsf{out}}$ given by the adversary is the same as the correct $(\mathsf{ct}_{\mathsf{out}}, \cdot, \cdot) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y)$, in which case the outputs of h and h' are identical and not $\perp$ by perfect correctness of Enc and Eval, as well as correctness of the delegation scheme.

   Specifically, considering the inputs to h, we know by correctness of delegation that, since $(\mathsf{ct}'_{\mathsf{out}}, \cdot, \cdot) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y)$, $\mathsf{ct}'_{\mathsf{out}} = g(\mathsf{ct}_x, \mathsf{ct}'_y) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, \mathsf{ct}_x, \mathsf{ct}'_y)$. Furthermore, by perfect correctness of the FHE scheme and the fact that ct_x and ct'_y are encryptions of x and y, respectively:

$$\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{ct}_{\mathsf{out}}) = \mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, \mathsf{ct}_x, \mathsf{ct}'_y)) = f(x, y')$$

   that is, the output of h will be identical to the output $f(x, y')$ of h'. In this case, $\mathcal{A}_{\mathsf{Del}}$ will produce output $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$ which is *rejected* because $\mathsf{ct}'_{\mathsf{out}} = \mathsf{ct}_{\mathsf{out}}$.

4. Steps (1) and (3) succeed, and $\mathsf{ct}'_{\mathsf{out}}$ given by the adversary is *not* the same as the correct $(\mathsf{ct}_{\mathsf{out}}, \cdot, \cdot) = \mathsf{Comp}_{\mathsf{Del}}(\mathsf{pk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y)$, in which case the outputs of h and h' *may be different* (and $\mathcal{A}_{\mathsf{Del}}$ will produce output $(g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}_{\mathsf{out}}, \mathsf{ct}'_{\mathsf{out}}, \pi_{\mathsf{Del}}, \pi'_{\mathsf{Del}}, 1^T, 1^{T'})$ which is *accepted* because $\mathsf{ct}'_{\mathsf{out}} \neq \mathsf{ct}_{\mathsf{out}}$ and $(\mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, 1^{T'})$ verifies successfully).

The above implies that the probability over possible interactions that the outputs of h and h' are different—which, as we have argued above, is equal to the statistical distance between the distributions $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$—is no greater[9] than the probability with which $\mathcal{A}_{\mathsf{Del}}$'s output is accepted. In particular, by our assumption that, for infinitely many $n \in \mathbb{N}$, there were x, y, z such that this statistical distance was greater than $1/p(n)$, this implies that the probability that $\mathcal{A}_{\mathsf{Del}}$'s output is accepted (for the corresponding inputs) must be greater than $1/p(n)$ for infinitely many $n \in \mathbb{N}$. But this contradicts the soundness of delegation, so the claim is proven. □

Now let $\Pi'_{h'}$ be identical to $\Pi_{h'}$, with the sole exception that the receiver's first-round message to the sender replaces the correctly generated $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$ with the corresponding encryption $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, 0; r_{\mathsf{Enc}(x)})$ of 0. We present the following claim comparing $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$:

Claim 9. *For any polynomial-time non-uniform distinguisher D, there exists negligible $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$ and inputs x, y, z, the distributions $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ cannot be distinguished by D with probability greater than $\epsilon(n)$.*

Proof. Intuitively, this follows from the CPA-security of the FHE scheme with respect to $T(n) \cdot \mathsf{poly}(n)$-time adversaries and the fact that both h' and the view of $\mathcal{A}'$ are independent of r_{FHE} and $\mathsf{sk}_{\mathsf{FHE}}$.

Formally, assume for contradiction that there exists a non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there are inputs x, y, z such that D is able to distinguish $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ with probability $1/p(n)$. We define a tuple of $T(n) \cdot \mathsf{poly}(n)$-time algorithms $(\mathcal{A}_{\mathsf{FHE}}, D')$ that can break the CPA-security of the FHE scheme $(\mathsf{Gen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}})$ with probability $1/p(n)$ as follows:

- $\mathcal{A}_{\mathsf{FHE}}$, on input 1^n, outputs $(0, x)$.
- D', on input $(1^n, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{ct}_x)$, where c is either $\mathsf{ct}^0_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, 0)$ or $\mathsf{ct}^1_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x)$, does the following:
  1. Generate $r_{\mathsf{Del}} \leftarrow \{0,1\}^*$ and $(\mathsf{pk}_{\mathsf{Del}}, \mathsf{sk}_{\mathsf{Del}}) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$.
  2. Run the corrupted sender $\mathcal{A}'$ with sender input y, auxiliary input z, and first-round message $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$. $\mathcal{A}'$ will generate a message $(y', \mathsf{pk}'_{\mathsf{FHE}}, \mathsf{pk}'_{\mathsf{Del}}, \mathsf{ct}'_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, r'_{\mathsf{Enc}(y)}, T')$ to send to $\mathcal{T}_{h'}$ and output $\mathsf{out}_{\mathcal{A}'}$. Store $\mathsf{out}_{\mathcal{A}'}$.
  3. Verify the following and set $\mathsf{out}_R = \perp$ if any are false. Otherwise, set $\mathsf{out}_R = f(x, y')$.
     (a) $\mathsf{pk}_{\mathsf{FHE}} = \mathsf{pk}'_{\mathsf{FHE}}$, $\mathsf{pk}_{\mathsf{Del}} = \mathsf{pk}'_{\mathsf{Del}}$, $\mathsf{ct}_x = \mathsf{ct}'_x$

[9]Note that equality is not guaranteed, as h could possibly accept a ciphertext $\mathsf{ct}'_{\mathsf{out}} \neq \mathsf{ct}_{\mathsf{out}}$ that still decrypts to $f(x, y)$.

(b) $\mathsf{ct}'_y = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, y'; r'_{\mathsf{Enc}(y)})$
(c) $\mathsf{Ver}_{\mathsf{Del}}(\mathsf{sk}_{\mathsf{Del}}, g, \mathsf{ct}_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, T') = \mathsf{Accept}$
for the functionality $g(c_1, c_2) = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, f, c_1, c_2)$

4. Return $D(1^n, (\mathsf{out}_{\mathcal{A}'}, \mathsf{out}_R))$.

First, notice that (given that the inputs $\mathsf{pk}_{\mathsf{FHE}}$ and $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, m)$ for either $m = 0$ or $m = x$ are generated correctly) the inputs to $\mathcal{A}'$ in step (2) of D' are identically distributed to either the inputs in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ (if $m = x$) or the inputs in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ (if $m = 0$). Hence, the view of $\mathcal{A}'$ in D' is identically distributed to the corresponding view in the respective experiment, which implies that the output $\mathsf{out}_{\mathcal{A}'}$ must be as well, as must the message sent to $\mathcal{T}_{h'}$.

It remains to argue about the receiver's output out_R; recall that the honest receiver's output in either experiment is given by the output of the ideal functionality $\mathcal{T}_{h'}$. However, out_R as defined in step (3) of D' can easily be seen to be identically distributed to the output of h' in the respective experiment $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ (if $m = x$) or $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ (if $m = 0$). This holds because, since R is honest, R's inputs $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ are honestly generated and so the verifications in steps (3a) and (3b) are identical to the respective checks in step (1) of h. Furthermore, the verification in step (3c) of D' is identical to the verification in step (3) of h, so it follows that $\mathsf{out}_R = \perp$ exactly when h' in the respective experiment would return $\perp$, and that, otherwise, $\mathsf{out}_R = f(x, y')$, which by the definition of h' is identical to what h' would return if not outputting $\perp$.

So we have argued that the distribution $(\mathsf{out}_{\mathcal{A}'}, \mathsf{out}_R)$ is identical to $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ when $m = x$ and to $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ when $m = 0$. But we have assumed that for infinitely many $n \in \mathbb{N}$ there exist x, y, z so that D can distinguish $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ with probability $1/p(n)$, i.e., that there is at least a $1/p(n)$ difference between the probability that $D(1^n, (\mathsf{out}_{\mathcal{A}'}, \mathsf{out}_R))$ returns 1 in the $m = x$ case and the respective probability in the $m = 0$ case. But, since D' returns precisely $D(1^n, (\mathsf{out}_{\mathcal{A}'}, \mathsf{out}_R))$, this gives us

$$|\Pr[D(1^n, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, 0)) = 1] - \Pr[D(1^n, \mathsf{pk}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x)) = 1]| \geq 1/p(n)$$

which, since $\mathcal{A}_{\mathsf{FHE}}$ always returns $(0, x)$, means that $(\mathcal{A}_{\mathsf{FHE}}, D')$ is able to break the CPA-security of the underlying FHE scheme (w.r.t. $T(n) \cdot \mathsf{poly}(n)$-time adversaries) with probability $1/p(n)$ for infinitely many $n \in \mathbb{N}$, a contradiction. $\square$

It remains to compare $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n, x, y, z)$; however, we claim that in fact these distributions are already identical. First, observe that the input provided to $\mathcal{A}'$ in $\mathcal{S}_S$ is identically distributed to the input provided to $\mathcal{A}'$ in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$; in both cases this consists of an honestly generated $\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x$ such that ct_x is the respective encryption of 0. So it follows that the adversary's output, as well as the message sent by the adversary to the ideal functionality, must be identically distributed between the two experiments. Demonstrating that the receiver's outputs are identical—that is, that the output of h' in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ is always equal to the output $f(x, y)$ in $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n, x, y, z)$—will follow from the following claim, to which we have already alluded in the previous two reductions:

Claim 10. *If R is honest, then, given messages $(x, \mathsf{pk}_{\mathsf{FHE}}, r_{\mathsf{Del}}, r_{\mathsf{Enc}})$ sent to $\mathcal{T}_{h'}$, $(\mathsf{pk}_{\mathsf{FHE}}, \mathsf{pk}_{\mathsf{Del}}, \mathsf{ct}_x)$ sent to $\mathcal{A}'$, and $(y', \mathsf{pk}'_{\mathsf{FHE}}, \mathsf{pk}'_{\mathsf{Del}}, \mathsf{ct}'_x, \mathsf{ct}'_y, \mathsf{ct}'_{\mathsf{out}}, \pi'_{\mathsf{Del}}, r'_{\mathsf{Enc}(y)}, T')$ sent by $\mathcal{A}'$ to $\mathcal{T}_{h'}$, the checks in step (2) of $\mathcal{S}_S$ succeed if and only if all checks in steps (1) and (3) of the functionality h' succeed.*

Proof. If R is honest, it must be the case that $(\mathsf{pk}_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$ and $\mathsf{ct}_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_{\mathsf{FHE}}, x ; r_{\mathsf{Enc}(x)})$; hence step (2a) of $\mathcal{S}_S$ is equivalent to verifying $\mathsf{pk}'_{\mathsf{FHE}} = \mathsf{pk}_{\mathsf{FHE}}$, $(\mathsf{pk}'_{\mathsf{Del}}, \cdot) = \mathsf{Gen}_{\mathsf{Del}}(1^\lambda; r_{\mathsf{Del}})$, and $\mathsf{ct}'_x = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}'_{\mathsf{FHE}}, x; r_{\mathsf{Enc}(x)})$, i.e., the first three checks of step (1) of h'. Step (2b) is trivially equivalent to the last check in step (1) of h' and step (2c) is trivially equivalent to the check in step (3) of h', completing the argument. □

This implies that the receiver in $\mathsf{Exec}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ will return $\perp$ as the output from h' precisely when $\mathcal{S}_S$ will return $\perp$ to the ideal functionality (based on the checks in step (2)) and cause the receiver in $\mathsf{Exec}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n, x, y, z)$ to return $\perp$. However, when $\mathcal{T}_f$ does not output $\perp$, it will always output $f(x, y')$ on the respective inputs x from the honest receiver and y' from $\mathcal{S}_S$; similarly, when $\mathcal{T}_{h'}$ does not return $\perp$, it will, by definition, *also* always output $f(x, y')$ on the respective input x from the honest receiver and y' from $\mathcal{A}'$. The above, then, is sufficient to conclude that the distributions $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S}_S,S}(1^n, x, y, z)$ are identical.

We conclude the proof of the lemma with a standard hybrid argument; specifically, if there exists some non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there are inputs x, y, z so that D can distinguish $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},S}(1^n, x, y, z)$ with probability $1/p(n)$, then D must likewise be able to distinguish one of the following pairs with probability $1/p'(n)$ for some polynomial $p'(\cdot)$:

- $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$
- $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$
- $\mathsf{Out}^{\mathcal{T}_{h'}}_{\Pi'_{h'},\mathcal{A}',S}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},S}(1^n, x, y, z)$

The first case would contradict Claim 8, the second case would contradict Claim 9, and the third case is impossible because we showed the distributions to be identical. Therefore, such a distinguisher D cannot exist. □

By the same logic, a standard hybrid argument shows that Lemmas 2 and 3 imply Theorem 7: if there were some non-uniform polynomial-time distinguisher D and polynomial $p(\cdot)$ such that, for infinitely many $n \in \mathbb{N}$, there were inputs x, y, z so that D could distinguish $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$ with probability $1/p(n)$, then D would be able to distinguish either:

- $\mathsf{Out}_{\Pi,\mathcal{A},I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$, or
- $\mathsf{Out}^{\mathcal{T}_h}_{\Pi_h,\mathcal{A}',I}(1^n, x, y, z)$ and $\mathsf{Out}^{\mathcal{T}_f}_{\Pi_f,\mathcal{S},I}(1^n, x, y, z)$

with probability $1/p'(n)$ for some polynomial $p'(\cdot)$. The first case would contradict Lemma 2 and the second Lemma 3; hence, Theorem 7 is proven.

References

[AEKP19] Gilad Asharov, Naomi Ephraim, Ilan Komargodski, and Rafael Pass. On perfect correctness without derandomization. Cryptology ePrint Archive, Report 2019/1025, 2019. `https://eprint.iacr.org/2019/1025`.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15:115–162, 01 2006.

[AMPR14] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404. Springer, Heidelberg, May 2014.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 111–120. ACM Press, June 2013.

[BCI$^+$13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In David B. Shmoys, editor, *46th ACM STOC*, pages 505–514. ACM Press, May / June 2014.

[BD18] Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 370–390. Springer, Heidelberg, November 2018.

[BGI$^+$17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 275–303. Springer, Heidelberg, December 2017.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

[BHHI10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 423–444. Springer, Heidelberg, May / June 2010.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th ACM STOC*, pages 474–482. ACM Press, June 2017.

[BJOV18] Saikrishna Badrinarayanan, Abhishek Jain, Rafail Ostrovsky, and Ivan Visconti. Non-interactive secure computation from one-way functions. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 118–138. Springer, Heidelberg, December 2018.

[BK18] Zvika Brakerski and Yael Tauman Kalai. Monotone batch np-delegation with applications to access control. Cryptology ePrint Archive, Report 2018/375, 2018. `https://eprint.iacr.org/2018/375`.

[BP04] Boaz Barak and Rafael Pass. On the possibility of one-message weak zero-knowledge. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 121–132. Springer, Heidelberg, February 2004.

[BP15] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 236–261. Springer, Heidelberg, November / December 2015.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CDG$^+$17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 33–65. Springer, Heidelberg, August 2017.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014.

[DLN$^+$04] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct Proofs for NP and Spooky Interactions. 2004.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[GIS$^+$10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 308–326. Springer, Heidelberg, February 2010.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of computer and system sciences*, 28(2):270–299, 1984.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010.

[Hai08] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 412–426. Springer, Heidelberg, March 2008.

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptology*, 25(1):158–193, 2012.

[HPV16] Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Composable security in the tamper-proof hardware model under minimal complexity. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 367–399. Springer, Heidelberg, October / November 2016.

[IKO$^+$11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 406–425. Springer, Heidelberg, May 2011.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, Heidelberg, August 2008.

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

[KMO89] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *30th FOCS*, pages 474–479. IEEE Computer Society Press, October / November 1989.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 335–354. Springer, Heidelberg, August 2004.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

[MR17] Payman Mohassel and Mike Rosulek. Non-interactive secure 2PC in the offline/online and batch settings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 425–455. Springer, Heidelberg, April / May 2017.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2003.

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: Achieving universal composability without trusted setup. In László Babai, editor, *36th ACM STOC*, pages 242–251. ACM Press, June 2004.

[Ps05] Rafael Pass and Abhi shelat. Unconditional characterizations of non-interactive zero-knowledge. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 118–134. Springer, Heidelberg, August 2005.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *59th FOCS*, pages 859–870. IEEE Computer Society Press, October 2018.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[SU11] Dominique Schröder and Dominique Unruh. Round optimal blind signatures. Cryptology ePrint Archive, Report 2011/264, 2011. `https://eprint.iacr.org/2011/264`.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

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

A Non-succinct NISC Protocols

Here, we state our candidate non-succinct NISC protocols for use in our main theorem. Specifically, while [BGI$^+$17] claims that NISC with superpolynomial-time simulation can be constructed from just a notion of "weak oblivious transfer" (see Appendix A.1 for a formal definition), their construction requires an *onto* one-way function, which cannot necessarily be constructed just from weak OT. While this distinction is unimportant in the context of their major results, we wish to show that our construction of succinct NISC can be based solely on subexponential LWE; hence, in Figure 4, we present an adaptation of their protocol which can be based solely on weak OT. The key difference is that, while [BGI$^+$17] relies on a witness-indistinguishable argument and the aforementioned onto one-way function, we instead discard the one-way function and rely on a two-round zero-knowledge argument with superpolynomial-time simulation, showing that this is sufficient via complexity leveraging and a careful sequence of hybrids.

Consider some quasi-polynomial functions $T_{\mathsf{OT}}(n)$, $T_{\mathsf{ExtOT}}(n)$, $T_{\mathsf{Sim}}(n)$, $T_{\mathsf{GC}}(n)$, $T_{\mathsf{ExtCom}}(n)$, $T_{\mathsf{Com}}(n)$ satisfying:

$$T_{\mathsf{GC}}(n) \gg T_{\mathsf{ExtOT}}(n) \gg T_{\mathsf{OT}}(n) \gg T_{\mathsf{ExtCom}}(n) \gg T_{\mathsf{Com}}(n) \gg T_{\mathsf{Sim}}(n)$$

where we write $T(n) \gg T'(n)$ to indicate that $T(n) > T'(n) \cdot p(n)$ for any polynomial $p(\cdot)$ and sufficiently large $n \in \mathbb{N}$. Specifically, we can achieve quasi-polynomial simulation by letting each function be given by $n^{\log^c(n)}$ for some different constant c. Π will use the following primitives:

- Let $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ be a 1-of-2 oblivious transfer scheme satisfying the definition of "$(T_{\mathsf{OT}}(\cdot), T_{\mathsf{ExtOT}}(\cdot))$-weak OT" given in Appendix A.1 with respective $T_{\mathsf{ExtOT}}(n) \cdot \mathsf{poly}(n)$-time extractor $\mathsf{Ext}_{\mathsf{OT}}$ (for $T_{\mathsf{ExtOT}}(n) \gg T_{\mathsf{OT}}(n)$, as otherwise $\mathsf{Ext}_{\mathsf{OT}}$ would trivially break chooser's security).
- Let $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$ be a two-message zero-knowledge protocol with $T_{\mathsf{Sim}}(n) \cdot \mathsf{poly}(n)$-time simulation; let $\mathsf{Sim}_{\mathsf{ZK}}$ be the respective simulator. [BGI$^+$17] shows that such schemes can be constructed from the above notion of "weak OT".
- Let $(\mathsf{Garble}, \mathsf{Eval})$ be a garbled circuit scheme secure against $T_{\mathsf{GC}}(n) \cdot \mathsf{poly}(n)$-time adversaries which satisfies perfect correctness (see Appendix A.2); let $\mathsf{Sim}_{\mathsf{GC}}$ be the respective simulator in the definition of security.
- Let $(\mathsf{Setup}_{\mathsf{Com}}, \mathsf{Com})$ be a secure two-round commitment scheme with $T_{\mathsf{ExtCom}}(n) \cdot \mathsf{poly}(n)$-time extraction which satisfies statistical binding and hiding against $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time adversaries (where $T_{\mathsf{ExtCom}}(n) \gg T_{\mathsf{Com}}(n)$, as otherwise $\mathsf{Ext}_{\mathsf{Com}}$ would trivially break hiding); let $\mathsf{Ext}_{\mathsf{Com}}$ denote the respective extractor.

As we notice that there exist constructions of both garbled circuit schemes [BHHI10] and two-round commitment schemes [Nao91] based on one-way functions, and as (even semi-honest) OT implies one-way functions [Hai08], this yields the following theorem:

Theorem 8 (Adapted from [BGI$^+$17])**.** *Assuming the existence of weak oblivious transfer, then, for any functionality $f(\cdot, \cdot)$ computable by a polynomial-time Turing machine, there exists a non-interactive secure computation protocol Π for f with quasi-polynomial simulation.*

Furthermore, since weak OT can be based on subexponential LWE [BD18], we have as a corollary:

Corollary 2 (Theorem 5)**.** *Assuming subexponential hardness of the Learning With Errors assumption, then, for any functionality $f(\cdot,\cdot)$ computable by a polynomial-time Turing machine, there exists a non-interactive secure computation protocol Π for f with quasi-polynomial simulation.*

The proof of Theorem 8 follows analogously to the original in [BGI$^+$17]; we provide a slightly abbreviated version here.

Proof. We separate into two cases based on whether the sender or receiver is corrupted.

Corrupted Sender. In the case where the sender is corrupted, we present the simulator given in Figure 5, which will run in time $T_{\mathsf{ExtCom}}(n) \cdot \mathsf{poly}(n) = \mathsf{poly}(n) n^{\log^c(n)}$ for constant c. It suffices to show that, between the real and ideal experiments, the outputs of the adversary and of the honest receiver (which simply outputs the result of the ideal functionality) are indistinguishable.

First, consider a hybrid experiment H where R, after receiving the malicious sender's second-round message, computes y^* by using $\mathsf{Ext}_{\mathsf{Com}}$ after verifying zk_2 and then computes and outputs $f(x, y^*)$, rather than computing an output by evaluating GC as in the real experiment. We claim that R's and the adversary's outputs are statistically close between the real experiment and H. The adversary's output is trivially identical as their view does not change. To reason about R's output in H, notice that the two experiments return $\perp$ under precisely the same conditions; furthermore, by security of the underlying ZK protocol, the experiments, with overwhelming probability, return something besides $\perp$ if and only if $(\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y) \in L$.

If $(\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y) \in L$, there exists a witness $w = (\overrightarrow{\mathsf{ot}}_1, \overrightarrow{K}, \overrightarrow{r_{\mathsf{OT}}}, r_{\mathsf{GC}}, c_1, y, r_{\mathsf{Com}})$ demonstrating that $(\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y)$ is properly generated given some randomness $(\overrightarrow{r_{\mathsf{OT}}}, r_{\mathsf{GC}}, r_{\mathsf{Com}})$ and the malicious sender's input y. In the real protocol, we know that, by *perfect* correctness of the OT protocol, the honest receiver will always recover the keys corresponding to their input x when $\overrightarrow{\mathsf{ot}}_2$ is correctly generated. Furthermore, by *perfect* correctness of the garbled circuit scheme, using the respective keys on a (correctly formed) garbled circuit GC will always result in the correct output $C(x) = f(x, y)$, where y is as given in the witness w. However, by statistical binding and extractability of Com, this y must with overwhelming probability (i.e., unless binding fails) be the same one as the y^* returned by $\mathsf{Ext}_{\mathsf{Com}}(c_y)$, and so in this case the output $C(x) = f(x, y)$ in the real experiment must always be equal to the output $f(x, y^*)$ of the ideal functionality in the ideal experiment, since the honest receiver will always provide the same x. Thus, the outputs of R and the adversary are statistically close between the real experiment and H whenever $(\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y) \in L$, implying that they are statistically close overall (since it is overwhelmingly likely that either $(\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y) \in L$ or both protocols return $\perp$).

So it suffices to compare H to the ideal experiment. The only difference between H and the ideal experiment is the way in which the message $\overrightarrow{\mathsf{ot}}_1$ is generated; hence, by $T_{\mathsf{OT}}(n)$-chooser's security of the OT protocol (see Appendix A.1), we can show that the joint distribution of the adversary's and receiver's output is indistinguishable between H and the ideal experiment. Specifically, assume for the sake of contradiction that there exists some non-uniform polynomial-time distinguisher D that can successfully distinguish between the respective joint distributions; we can construct a $T_{\mathsf{ExtCom}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D' that breaks chooser's security of the underlying OT scheme (note that the OT is assumed secure against $T_{\mathsf{OT}}(n) \cdot \mathsf{poly}(n)$-time adversaries and $T_{\mathsf{OT}}(n) \gg T_{\mathsf{ExtCom}}(n)$) by, given some OT message ot_1^*, running the receiver protocol and respective adversarial sender in the experiment using ot_1^* (note that the experiment must run $\mathsf{Ext}_{\mathsf{Com}}$, hence the running time bound) to generate the respective output distribution and subsequently returning the output

Input: The receiver R and the sender S are given input $x, y \in \{0,1\}^n$, respectively, and both parties have common input 1^n.
Output: R receives $f(x,y)$.

Round 1: R proceeds as follows:

1. Generate $\overrightarrow{\mathsf{ot}}_1, \overrightarrow{\sigma_{\mathsf{OT}}}$ by, for $i \in [n]$, taking $((\overrightarrow{\mathsf{ot}}_1)_i, (\overrightarrow{\sigma_{\mathsf{OT}}})_i) \leftarrow \mathsf{OT}_1(1^n, x_i)$.
2. Generate $(\mathsf{zk}_1, \sigma_{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_1(1^n)$.
3. Generate $c_1 \leftarrow \mathsf{Setup}_{\mathsf{Com}}(1^n)$.
4. Send $(\overrightarrow{\mathsf{ot}}_1, \mathsf{zk}_1, c_1)$ to S.

Round 2: S proceeds as follows:

1. Compute $c_y = \mathsf{Com}(c_1, y; r_{\mathsf{Com}})$ using randomness $r_{\mathsf{Com}} \leftarrow \{0,1\}^*$.
2. Generate keys $\overrightarrow{K} = \{K_{i,b}\}_{i\in[n], b\in\{0,1\}}$, where $K_{i,b} \leftarrow \{0,1\}^n$ for each $i \in [n]$ and $b \in \{0,1\}$.
3. Generate garbled circuit $\mathsf{GC} = \mathsf{Garble}(\overrightarrow{K}, C; r_{\mathsf{GC}})$, where $C(\cdot)$ is the circuit that computes $f(\cdot, y)$.
4. Compute $\overrightarrow{\mathsf{ot}}_2$ and $\overrightarrow{r_{\mathsf{OT}}}$ by, for $i \in [n]$, taking $(\overrightarrow{r_{\mathsf{OT}}})_i \leftarrow \{0,1\}^*$ and $(\overrightarrow{\mathsf{ot}}_2)_i = \mathsf{OT}_2((\overrightarrow{\mathsf{ot}}_1)_i, K_{i,0}, K_{i_1}; (\overrightarrow{r_{\mathsf{OT}}})_i)$.
5. Compute $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y), (\overrightarrow{\mathsf{ot}}_1, \overrightarrow{K}, \overrightarrow{r_{\mathsf{OT}}}, r_{\mathsf{GC}}, c_1, y, r_{\mathsf{Com}}))$ for the language L consisting of tuples $x = (\mathsf{ot}_2, \mathsf{GC}, c_y)$ such that there exists a witness $w = (\overrightarrow{\mathsf{ot}}_1, \overrightarrow{K}, \overrightarrow{r_{\mathsf{OT}}}, r_{\mathsf{GC}}, c_1, y, r_{\mathsf{Com}})$ satisfying:
   (a) $(\overrightarrow{\mathsf{ot}}_2)_i = \mathsf{OT}_2((\overrightarrow{\mathsf{ot}}_1)_i, K_{i,0}, K_{i,1}; (\overrightarrow{r_{\mathsf{OT}}})_i)$ for all $i \in [n]$.
   (b) $\mathsf{GC} = \mathsf{Garble}(\overrightarrow{K}, C; r_{\mathsf{GC}})$, where $C(\cdot)$ is the circuit that computes $f(\cdot, y)$.
   (c) $c_y = \mathsf{Com}(c_1, y; r_{\mathsf{Com}})$.
6. Send $(\mathsf{zk}_2, \overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y)$ to R.

Output phase: R proceeds as follows:

1. Verify that $\mathsf{ZK}_3(\mathsf{zk}_2, (\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y), \sigma_{\mathsf{ZK}}) = \mathsf{Accept}$; if not, output $\perp$.
2. Compute $\overrightarrow{K^*}$ by taking $K_i^* = \mathsf{OT}_3((\overrightarrow{\mathsf{ot}}_2)_i, (\overrightarrow{\sigma_{\mathsf{OT}}})_i)$ for each $i \in [n]$.
3. Output $\mathsf{Eval}(\overrightarrow{K^*}, \mathsf{GC})$.

Figure 4: Protocol Π for (non-succinct) non-interactive secure computation.

Simulator $\mathcal{S}_S$

1. Generate $\overrightarrow{\mathsf{ot}}_1, \overrightarrow{\sigma_{\mathsf{OT}}}$ by, for $i \in [n]$, taking $((\overrightarrow{\mathsf{ot}}_1)_i, (\overrightarrow{\sigma_{\mathsf{OT}}})_i) \leftarrow \mathsf{OT}_1(1^n, 0)$.
2. Generate $(\mathsf{zk}_1, \sigma_{\mathsf{ZK}}) \leftarrow \mathsf{ZK}_1(1^n)$.
3. Generate $c_1 \leftarrow \mathsf{Setup}_{\mathsf{Com}}(1^n)$.
4. Send $(\overrightarrow{\mathsf{ot}}_1, \mathsf{zk}_1, c_1)$ to the (malicious) sender S.
5. Receive output $(\mathsf{zk}_2, \overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y)$ from S.
6. Verify that $\mathsf{ZK}_3(\mathsf{zk}_2, (\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y), \sigma_{\mathsf{ZK}}) = \mathsf{Accept}$; if not, output $\perp$.
7. Use the extractor $\mathsf{Ext}_{\mathsf{Com}}$ to compute $y^* \leftarrow \mathsf{Ext}_{\mathsf{Com}}(c_y)$, and send y^* to the ideal functionality $\mathcal{T}_f$.

Figure 5: Simulator for a malicious sender.

of D on the distribution. Since the experiments are identical aside from the first OT message, D being able to distinguish the ideal experiment (with OT input 0) from H (with a non-zero OT input) would imply D' successfully distinguishing between $\mathsf{OT}_1(1^n, 0)$ and $\mathsf{OT}_1(1^n, 1)$, hence contradicting the OT's security and completing the argument for the malicious sender case.

Corrupted Receiver. Otherwise, if the receiver is corrupted, we use the $T_{\mathsf{ExtOT}}(n) \cdot \mathsf{poly}(n)$-time simulator given in Figure 6; once again we note that the running time is given by $\mathsf{poly}(n) n^{\log^c(n)}$ for constant c. It suffices to show that the output of the malicious receiver is indistinguishable between the two experiments, which we can do by showing that the views of the adversary R are likewise indistinguishable.

To argue that this is the case, we introduce the following sequence of hybrids:

1. Let hybrid H_0 denote the real experiment.

2. Let H_1 be identical to H_0, except that zk_2, rather than being computed honestly by ZK_2, is computed as $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y))$.

3. Let H_2 be identical to H_1, except that the commitment c_y is generated as $c_y = \mathsf{Com}(c_1, 0)$ rather than $\mathsf{Com}(c_1, y)$.

4. Let H_3 be identical to H_2, except that H_3 now uses the extractor $\mathsf{Ext}_{\mathsf{OT}}$ to recover x^* and generates the inputs $K_{i,b}$ to OT_2 (for any i, b such that $x_i^* \neq b$) uniformly at random rather than using the respective keys.

5. Let H_4 be identical to H_3, except that H_4 additionally computes $z = f(x^*, y)$, and furthermore the garbled circuit GC, rather than being computed honestly by Garble, is generated as $\mathsf{GC} = \mathsf{Sim}_{\mathsf{GC}}(\overrightarrow{K}, z)$.

Notice that H_4 is now identical to the behavior of $\mathcal{S}_R$ in the ideal world; hence, it suffices to argue that the above sequence of hybrids are indistinguishable by the non-uniform polynomial-time adversary $\mathcal{A}$ in order to show that the adversary's view (and hence their output) must likewise be indistinguishable between the real and ideal experiments.

H_0 is (non-uniform polynomial-time) indistinguishable from H_1 by simulation-based security of the zero-knowledge protocol $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$: since $(\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y) \in L$ (i.e., they are honestly generated, and hence there exists a valid witness w), the definition of security implies that $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y))$ and $\mathsf{zk}_2 \leftarrow \mathsf{ZK}_2(1^n, \mathsf{zk}_1, (\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y), w)$ (for some valid witness w) are indistinguishable by any non-uniform polynomial-time adversary.

```
Simulator $\mathcal{S}_R$
1. Run the malicious receiver $R$ to receive a first-round message $(\overrightarrow{\mathsf{ot}}_1, \mathsf{zk}_1, c_1)$.
2. Use the extractor $\mathsf{Ext}_{\mathsf{OT}}$ to compute $x^*$ by taking $x_i^* \leftarrow \mathsf{Ext}_{\mathsf{OT}}((\overrightarrow{\mathsf{ot}}_1)_i)$ for each $i \in [n]$.
3. Send $x^*$ to the ideal functionality $\mathcal{T}_f$ and receive an output $z$.
4. Compute $c_y = \mathsf{Com}(c_1, 0)$.
5. Generate keys $\overrightarrow{K} = \{K_{i,b}\}_{i\in[n],b\in\{0,1\}}$, where $K_{i,b} \leftarrow \{0,1\}^n$ for each $i \in [n]$ and $b \in \{0,1\}$.
6. Generate garbled circuit $\mathsf{GC} = \mathsf{Sim}_{\mathsf{GC}}(\overrightarrow{K}, z)$.
7. Compute $\overrightarrow{\mathsf{ot}}_2$ by, for $i \in [n]$, taking $(\overrightarrow{\mathsf{ot}}_2)_i = \mathsf{OT}_2((\overrightarrow{\mathsf{ot}}_1)_i, K_{i,0}, K_{i,1})$.
8. Compute $\mathsf{zk}_2 \leftarrow \mathsf{Sim}_{\mathsf{ZK}}(\mathsf{zk}_1, (\overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y))$.
9. Send $(\mathsf{zk}_2, \overrightarrow{\mathsf{ot}}_2, \mathsf{GC}, c_y)$ to $R$.
```

Figure 6: Simulator for a malicious receiver.

H_2 is indistinguishable from H_1 by the hiding property of the underlying commitment scheme Com (i.e., that, for any adversarially chosen first message c_1 and messages m_0, m_1, the respective commitments are indistinguishable by $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time adversaries). The only difference between H_1 and H_2 is how the commitment c_y is generated; hence, if there were a non-uniform polynomial-time distinguisher D able to distinguish the adversary's view between the experiments H_1 and H_2, then it would be possible to construct a $T_{\mathsf{Sim}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D' between $\mathsf{Com}(c_1, 0)$ and $\mathsf{Com}(c_1, y)$. Specifically, D' could run the adversary to obtain the first message c_1, request a commitment c_y of one of the two values (0 or y) under the message c_1, use c_y in place of the respective commitment while emulating the rest of the experiment in $T_{\mathsf{Sim}}(n) \cdot \mathsf{poly}(n)$ time, and return the output of D on the resulting view. If D distinguishes between the respective experiments H_1 and H_2, then D' would distinguish between $\mathsf{Com}(c_1, 0)$ and $\mathsf{Com}(c_1, y)$, contradicting the hiding of Com since we assumed it to have security against $T_{\mathsf{Com}}(n) \cdot \mathsf{poly}(n)$-time adversaries and $T_{\mathsf{Com}}(n) \gg T_{\mathsf{Sim}}(n)$. (Note that D' does indeed run in time $T_{\mathsf{Sim}}(n) \cdot \mathsf{poly}(n)$; while the final simulator $\mathcal{S}_R$ uses the extractor $\mathsf{Ext}_{\mathsf{OT}}$, which requires far more time, none of the hybrids before H_3 use this extractor.)

H_2 is indistinguishable from H_3 by sender's security of the weak OT protocol $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ (see Appendix A.1). Specifically, the only difference between the adversary's view in H_2 and H_3 is in how the OT message $\overrightarrow{\mathsf{ot}}_2$ is generated. If the adversary's view of the experiments were distinguishable by some non-uniform polynomial-time distinguisher D, it would be possible to create a $T_{\mathsf{ExtOT}}(n) \cdot \mathsf{poly}(n)$-time distinguisher D' (note that the running time is significantly higher than distinguishing H_1 and H_2 as we now need to run $\mathsf{Ext}_{\mathsf{OT}}$) between $\mathsf{OT}_2((\overrightarrow{\mathsf{ot}}_1)_i, K_{i,0}, K_{i,1})$ and OT_2 run on the same inputs but with a different $K_{i,b}$ corresponding to $b \neq \mathsf{Ext}_{\mathsf{OT}}((\overrightarrow{\mathsf{ot}}_1)_i)$, directly contradicting (statistical) sender's security of the OT. The distinguisher would use the respective OT_2 output in place of a randomly chosen one of the OT messages, emulate the rest of the experiment in $T_{\mathsf{ExtOT}}(n) \cdot \mathsf{poly}(n)$ time (specifically, in $T_{\mathsf{ExtOT}}(n) + T_{\mathsf{Sim}}(n) + \mathsf{poly}(n)$ time, since the experiment runs both $\mathsf{Sim}_{\mathsf{ZK}}$ and $\mathsf{Ext}_{\mathsf{OT}}$; however, we collect the terms due to the observation that $T_{\mathsf{Sim}}(n) \ll T_{\mathsf{ExtOT}}(n)$), and return the output of D on the resulting view. If D' were able to distinguish between the respective experiments, then D would be able to distinguish between the respective OT messages, breaking statistical sender's security.

Lastly, H_3 is indistinguishable from H_4 by security of the garbled circuit scheme $(\mathsf{Garble}, \mathsf{Eval})$ with respect to the simulator $\mathsf{Sim}_{\mathsf{GC}}$. The only difference between the adversary's views in H_3 and H_4 is in how the garbled circuit GC is generated, so, as before, if we had some non-uniform polynomial-time distinguisher D between the respective views in experiments H_3 and H_4, we could construct a $T_{\mathsf{ExtOT}}(n) \cdot \mathsf{poly}(n)$-time distinguisher between real and simulated garbled circuits by running the experiment while using $\mathsf{Ext_{OT}}$ to recover x^* and $z = f(x^*, y)$, obtaining a (real or simulated) garbled circuit for the respective inputs and output, substituting the circuit for GC, emulating the rest of the experiment (again, in time $T_{\mathsf{ExtOT}}(n) + T_{\mathsf{Sim}}(n) + \mathsf{poly}(n) = T_{\mathsf{ExtOT}}(n) \cdot \mathsf{poly}(n)$), and returning the output of D on the resulting view. If D successfully distinguishes between H_3 and H_4, then D' would in turn distinguish between the real and simulated circuits. This, however, would contradict security of $(\mathsf{Garble}, \mathsf{Eval})$, since we have assumed it secure against $T_{\mathsf{GC}}(n) \cdot \mathsf{poly}(n)$-time adversaries for $T_{\mathsf{GC}}(n) \gg T_{\mathsf{ExtOT}}(n)$. □

Extension to the CRS model. A NISC protocol from any UC-secure OT is claimed in ([IPS08], Appendix B), which can be instantiated with the OT protocol in the CRS model of [PVW08] to get a NISC protocol in the CRS model (with simulation based security). As their protocol is a bit complicated (and only informally analyzed), we point out that the above SPS protocol can be easily modified (and the proof directly extends) to give a NISC in the CRS model from any UC-secure OT in the CRS model, as follows:[10]

- We now require $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ to be a maliciously secure and *fully simulatable* oblivious transfer protocol (which, in the CRS model, can still be based on LWE [PVW08]), and we require $(\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$ to be a two-round zero-knowledge protocol with polynomial-time simulation in the CRS model, which is implied by OT [Kil88, KMO89, Ps05].
- We can relax the security on all underlying primitives to hold against polynomial-time adversaries, rather than $T(n) \cdot \mathsf{poly}(n)$-time adversaries.
- We remove the commitment scheme Com from the protocol (and remove c_1, c_y, r_{Com}, and the respective check from the language L) and let the simulators extract the inputs from the malicious party by using the respective (polynomial-time) simulators for the OT protocol instead of the superpolynomial-time extractors for weak OT and Com, so that the simulators will run in polynomial time.

Aside from these differences, the protocol and proof proceed identically to the above, yielding the following theorem:

Theorem 9. *Assuming polynomial security of the Learning With Errors assumption, then, for any functionality $f(\cdot, \cdot)$ computable by a polynomial-time Turing machine, there exists a non-interactive secure computation protocol Π for f with polynomial-time simulation in the CRS model.*

A.1 Definition of Weak Oblivious Transfer

For completeness, we present the definition of "weak OT" given in [BGI+17].

Definition 7 ([BGI+17]). *A two-round 1-of-2 **weak oblivious transfer protocol** is given by three algorithms, $(\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$, defining an interaction between a sender S and a chooser (receiver) R as follows:*

[10] Our protocol is simpler but this is only because we rely on the underlying OT protocol in a *non-black-box* way whereas [IPS08] relies on it only in a *black-box* way.

- $\mathsf{OT}_1(1^n, b) \to (\mathsf{ot}_1, \sigma)$ *generates R's message* ot_1 *and persistent state σ (which is not sent to S) given the security parameter n and R's choice bit $b \in \{0,1\}$.*
- $\mathsf{OT}_2(\mathsf{ot}_1, x_0, x_1) \to \mathsf{ot}_2$ *generates S's message* ot_2 *given S's messages x_0, x_1 and R's message* ot_1.
- $\mathsf{OT}_3(\mathsf{ot}_2, \sigma) \to x'_b$ *generates R's output x'_b given the state σ and S's message* ot_2.

such that the following properties hold:

1. ***Perfect correctness:*** *For any $n \in \mathbb{N}$, any choice bit $b \in \{0,1\}$, and any inputs $x_0, x_1 \in \{0,1\}^n$:*
$$Pr[(\mathsf{ot}_1, \sigma) \leftarrow \mathsf{OT}_1(1^n, b) : x_b = \mathsf{OT}_3(\sigma, \mathsf{OT}_2(x_0, x_1, \mathsf{ot}_1))] = 1$$
2. ***Chooser's security:*** *For any non-uniform polynomial-time distinguisher D, there exists a negligible $\epsilon(\cdot)$ such that, for any $n \in \mathbb{N}$:*
$$|Pr[D(\mathsf{OT}_1(1^n, 0)) = 1] - Pr[D(\mathsf{OT}_1(1^n, 1)) = 1]| < \epsilon(n)$$
*If the above holds with respect to $T(n) \cdot \mathsf{poly}(n)$-time distinguishers D, we say that the protocol satisfies $T(\cdot)$**-chooser's security**, or alternatively that it is $T(\cdot)$**-secure**.*
3. ***Statistical sender's security:*** *There exists an extractor* $\mathsf{Ext_{OT}}$ *that, on input* ot_1, *outputs 0 if $Pr[\mathsf{OT}_1(1^n, 0) = \mathsf{ot}_1] > 0$ and 1 otherwise, and, for any (unbounded) distinguisher D, there exists negligible $\epsilon(\cdot)$ such that, for any* ot_1 *and any K_0, K_1, K'_0, K'_1 for which $K_{\mathsf{Ext_{OT}}(\mathsf{ot}_1)} = K'_{\mathsf{Ext_{OT}}(\mathsf{ot}_1)}$:*
$$|Pr[D(\mathsf{OT}_2(\mathsf{ot}_1, K_0, K_1)) = 1] - Pr\left[D(\mathsf{OT}_2(\mathsf{ot}_1, K'_0, K'_1)) = 1\right]| < \epsilon(n)$$
*We say that the protocol is $T'(\cdot)$**-extractable** if there exists an* $\mathsf{Ext_{OT}}$ *satisfying the above that runs in time $T'(n) \cdot \mathsf{poly}(n)$.*

*Lastly, for some $T(\cdot) < T'(\cdot)$, we call a weak OT protocol a $(T(n), T'(n))$**-weak OT** protocol if it is $T(\cdot)$-secure and $T'(\cdot)$-extractable.*

We state the following theorem for the existence of weak OT:

Theorem 10 ([BD18])**.** *Assuming subexponential security of the Learning With Errors assumption, then, for every $c < c'$, there exists a $(T(\cdot), T'(\cdot))$-weak OT protocol for $T(n) = n^{log^c(n)}, T'(n) = n^{log^{c'}(n)}$.*

A.2 Definition of Garbled Circuit Schemes

Definition 8 (based on [Yao86, BHHI10])**.** *A **garbled circuit scheme** consists of a pair of algorithms* (Garble, Eval) *such that:*

- $\mathsf{Garble}(\overrightarrow{K}, C) \to \mathsf{GC}$*: given security parameter n, takes as input a circuit C with input size n and* labels $\overrightarrow{K} = \{K_{b,i}\}_{b \in \{0,1\}, i \in [n]}$ *corresponding to each assignment (0 or 1) for each input wire, and outputs a garbled circuit* GC.

- $\mathsf{Eval}(\overrightarrow{K^*}, \mathsf{GC}) \to y$*: takes as input a garbled circuit* GC *and a garbled input* x *represented by the corresponding labels* $\overrightarrow{K^*} = \{K_{x_i,i}\}_{i\in[n]}$*, and outputs* y*.*

We consider garbled circuit schemes having the following properties:

1. ***Perfect correctness:*** *For any security parameter* $n \in \mathbb{N}$*, any circuit* C *with input length* $[n]$*, any labels* $\overrightarrow{K} = \{K_{b,i}\}_{b\in\{0,1\},i\in[n]} \in (\{0,1\}^n)^{2n}$ *(such that* $K_{0,i} \neq K_{1,i}$ *for each* i*), and any input* $x = x_1||\dots||x_n \in \{0,1\}^n$*, letting* $\overrightarrow{K^*} = \{K_{x_i,i}\}_{i\in[n]}$*:*

$$\Pr[\mathsf{Eval}(\overrightarrow{K^*}, \mathsf{Garble}(\overrightarrow{K}, C)) = C(x)] = 1$$

2. ***Security:*** *There exists an efficient simulator* Sim *such that, for all non-uniform polynomial-time distinguishers* D*, there exists a negligible function* $\epsilon(\cdot)$ *such that, for all* $n \in \mathbb{N}$*, for any circuit* C *with input length* n *and any input* $x = x_1||\dots||x_n \in \{0,1\}^n$*, letting* $\overrightarrow{K} = \{K_{b,i}\}_{b\in\{0,1\},i\in[n]} \leftarrow (\{0,1\}^n)^{2n}$ *be uniformly random and* $\overrightarrow{K^*} = \{K_{x_i,i}\}_{i\in[n]}$*:*

$$|\Pr[D(\mathsf{Garble}(\overrightarrow{K}, C), \overrightarrow{K^*}) = 1] - \Pr[D(\mathsf{Sim}(\overrightarrow{K}, C(x)), \overrightarrow{K^*}) = 1]| < \epsilon(n)$$

   If the above holds for $T(n)\mathsf{poly}(n)$*-time distinguishers* D*, we say that the scheme is* $T(\cdot)$***-secure****.*

We state the following theorem for existence:

Theorem 11 ([AIK06, BHHI10])**.** *Assuming the existence of subexponentially secure one-way functions, then, for constant* c*, there exists a* $T(\cdot)$*-secure garbled circuit scheme satisfying perfect correctness for* $T(n) = n^{log^c(n)}$*.*