Maliciously-Secure MrNISC in the Plain Model

Rex Fernando* Aayush Jain† Ilan Komargodski‡

September 30, 2021

Abstract

In this work we study strong versions of round-optimal MPC. A recent work of Benhamouda and Lin (TCC '20) identified a version of secure multiparty computation (MPC), termed *Multiparty reusable Non-Interactive Secure Computation* (MrNISC), that combines at the same time several fundamental aspects of secure computation with standard simulation security into one primitive: round-optimality, succinctness, concurrency, and adaptivity. In more detail, MrNISC is essentially a two-round MPC protocol where the first round of messages serves as a reusable commitment to the private inputs of participating parties. Using these commitments, any subset of parties can later compute any function of their choice on their respective inputs by broadcasting one message each. Anyone who sees these parties' commitments and evaluation messages (even an outside observer) can learn the function output and nothing else. Importantly, the input commitments can be computed without knowing anything about other participating parties (neither their identities nor their number) and they are reusable across any number of computations.

By now, there are several known MrNISC protocols from either (bilinear) group-based assumptions or from LWE. They all satisfy semi-malicious security (in the plain model) and require trusted setup assumptions in order to get malicious security. We are interested in maliciously secure MrNISC protocols *in the plain model, without trusted setup*. Since the standard notion of polynomial simulation is un-achievable in less than four rounds, we focus on MrNISC with *super-polynomial*-time simulation (SPS).

Our main result is the first maliciously secure SPS MrNISC in the plain model. The result is obtained by generically compiling any semi-malicious MrNISC and the security of our compiler relies on several well-founded assumptions, including an indistinguishability obfuscator and a time-lock puzzle (all of which need to be sub-exponentially hard). As a special case we also obtain the first 2-round maliciously secure SPS MPC based on well-founded assumptions. This MPC is also concurrently self-composable and its first message is short (i.e., its size is independent of the number of the participating parties) and reusable throughout any number of computations.

*UCLA. Email: rex1fernando@gmail.com. Work done in part while the author was visiting Carnegie Mellon University.

†UCLA. Email: aayushjain1728@gmail.com.

‡Hebrew University and NTT Research. Email: ilank@cs.huji.ac.il. Supported in part by an Alon Young Faculty Fellowship and by an ISF grant (No. 1774/20).

Contents

1 Introduction

In this work, we study secure multiparty computation (MPC), a fundamental primitive in cryptography which allows mutually distrusting parties to securely compute any function on their respective inputs without revealing anything besides the output, even if some of the parties are acting maliciously. While feasibility results were discovered already in the late 80s [GMW87, BGW88, CCD88], there has been significant effort in obtaining constructions that are *better* in some aspect. One important criterion, which has received much study in the past decade, is round complexity. Other interesting criteria include achieving security under concurrent executions, and achieving reusability of some messages of the protocol across sessions.

A recent work by Benhamouda and Lin [BL20] identified a strong version of MPC, termed *Multiparty reusable Non-Interactive Secure Computation* (MrNISC), that combines at the same time several aspects of secure computation. Specifically, an MrNISC is a **round-optimal** MPC that satisfies the standard notion of simulation security and further satisfies natural and desirable properties, like **succinctness**, **concurrency**, **adaptivity**, and **reusability**. Each one of these criteria is by itself a fundamental aspect of secure computation that has received significant attention over the years. MrNISC captures all of these features at the same time by requiring the following general structure:

1. *Round 1*: each party broadcasts an encoding of their private input x_i. This is done independently of the function going to be computed, and independently of the total number of parties.

   Parties can join the system at any time by broadcasting their first round message.

2. *Round 2*: any subset I of parties can jointly compute a function f on their inputs $x_I = \{x_i\}_{i \in I}$ by broadcasting (each) a single public message. Any party who sees both rounds (even an outside observer) can learn $f(x_I)$ and nothing else.

   Importantly, the second round can be repeated arbitrarily many times with different functions and different sets of parties.

We briefly explain how an MrNISC is round-optimal, concurrent, adaptive-secure and reusable. It is well-known that completely non-interactive MPC cannot satisfy the standard simulation security notion, due to the so called "residual-function" attack [BGI+14]. Therefore, MrNISC is round-optimal. The round 1 message of each party is independent of the number of parties in the system, and so MrNISC is succinct. A single party can participate in multiple concurrent evaluation sessions and round 1 messages can appear at any point in time, therefore MrNISC is concurrent and adaptive. Indeed, security must hold even if adversary can adaptively add new parties to the game, either corrupt or honest, and can adaptively choose their inputs and new function evaluations. Finally, the round 1 message can be used for multiple evaluations, and so MrNISC is reusable.

There has been exciting progress with respect to building MrNISC protocols from standard assumptions. Benhamouda and Lin [BL20] constructed such a protocol for all efficiently computable functionalities relying on the SXDH assumption in asymmetric bilinear groups. In two concurrent follow-up works, Ananth et al. [AJJM21] and Benhamouda et al. [BJKL21] obtained MrNISC protocols relying on Learning With Errors (LWE). All three papers achieve protocols with *semi-malicious* security, a slight strengthening of semi-honest security wherein the adversary is allowed to choose its random tape arbitrarily, but otherwise must follow the prescribed protocol. The work of [BL20] also showed how to achieve *malicious* security, where the protocol must be private and resilient even in the presence of parties which deviate arbitrarily from the protocol specification.

They do this using a trusted setup (i.e., a common random string), using standard techniques for transforming semi-malicious protocols into malicious ones.

Given this state of affairs, a very natural question is the following:

Is there a maliciously secure MrNISC in the plain model (without trusted setup)?

At first glance, it seems that a positive answer to this question would contradict known lower bounds. The work of [KO04] showed that four rounds of communication are necessary even for secure two-party computation, assuming a black-box simulator. Furthermore, even non-black-box simulation is impossible in two rounds if the simulator runs in polynomial time [GO94].

We would like to avoid these lower bounds, and so we turn to the notion of *super-polynomial time simulation.* Super-polynomial time simulation (SPS) [Pas03, PS04] is the widely accepted standard relaxation to get around the above impossibility results for round-efficient protocols. In the SPS setting, we say that for any polynomial-time adversary there should exist an ideal-world simulator that runs in super-polynomial time, such that the two worlds are indistinguishable. What this means is that no polynomial-time adversary in the real world can learn any more than what a super-polynomial-time machine could learn in the ideal world.

Several recent works explore the round complexity of MPC in the SPS setting. To put the question of SPS MrNISC in context, we discuss them here. The works of Badrinarayanan et al. [BGI+17] and Jain et al. [JKKR17] construct two-round two-party computation protocols satisfying malicious SPS security in the plain model. Badrinarayanan et al. [BGJ+17] constructed a three-round maliciously secure SPS MPC. Morgan et al. [MPP20] built a succinct two-round two-party maliciously-secure SPS protocol with only one party getting the output with the same security as above. However, the question of two-round MPC in the SPS setting has remained open for several years.

Up until a few weeks ago, we did not have a general *two*-round MPC in the plain model, from any assumption. This changed with the very recent exciting work[1] of Agarwal, Bartusek, Goyal, Khurana, and Malavolta [ABG+21] giving the first two-round maliciously secure SPS MPC (in the plain model). Their construction relies on a host of (sub-exponentially strong) standard assumptions plus a special type of *non-interactive* non-malleable commitment. Unfortunately, the only known instantiations of the latter rely on strong and non-standard assumptions. One instantiation relies on factoring-based *adaptive* one-way functions [PPV08],[2] a highly non-standard and non-falsifiable assumption which incorporates a strong non-malleability flavor. Another instantiation relies on *keyless* multi-collision resistant hash functions [BKP18] and an exponential variant of the "hardness amplifiability" assumption of [BL18]. While both of these assumptions are (sub-exponentially) falsifiable, they are still highly non-standard:

1. A keyless multi-collision resistant hash function is a single publicly known function for which (roughly) collisions are "incompressible", namely, it is impossible to encode significantly more than k collisions using only k bits of information. While keyless hash functions are formally a plain-model assumption, there is no known plain-model instantiation based on standard assumptions. The only known instantiation is either in the random oracle model, or by heuristically assuming that some cryptographic hash function, like SHA-256, is such.

[1] The paper was accepted to TCC 2021 but it only became publicly available on ePrint on September 20, 2021. Our work was done independently of theirs.

[2] An adaptive one-way function is a non-falsifiable hardness assumption postulating the existence of a one-way function f that is hard to invert on a random point $y = f(x)$ even if you get access to an inversion oracle that inverts it on every other point $y' \neq y$.

2. Hardness amplification assumptions postulate (roughly) that the XOR of independently committed random bits cannot be predicted with sufficiently large advantage. There are concrete (contrived) counter examples for this type of assumptions showing that they are generically false [DJMW12], although they certainly might hold for specific constructions.

   The specific variant used by Agarwal et al. is novel to their work. It assumes *exponential* hardness amplification against PPT adversaries, i.e., that there exists a constant $\delta > 0$ such that for large enough ℓ, the XOR of ℓ independently committed random bits cannot be predicted by a PPT adversary with advantage better than $2^{-\ell\delta}$. This assumption (similarly to [PPV08]'s adaptive one-way functions) also incorporates a non-malleability flavor.

This state of affairs leaves us with a large gap in our understanding: there is no known plain-model two-round maliciously secure SPS MPC (let alone MrNISC!) based on well-founded assumptions.

1.1 Our Results

In this work we give the first affirmative answer to the above question and close the gap in our understanding. Specifically, relying on well-founded assumptions, we obtain a maliciously secure SPS MrNISC in the plain model without any trusted setup. Our construction is a generic transformation from any semi-malicious secure MrNISC, and it further relies on the quantum hardness of Learning With Errors (LWE) and the classical hardness of SXDH, as well as on the existence of an indistinguishability obfuscator iO for polynomial-size circuits, and a time-lock puzzle.

Theorem 1 (Main Result)**.** *There exists a generic compiler that starts off with any semi-maliciously secure MrNISC and turns it into a maliciously secure SPS MrNISC in the plain model. Security of the construction relies on the quantum hardness of LWE, classical hardness of SXDH, the existence of a (classically hard) indistinguishability obfuscation scheme, and a (classically hard but quantum broken) time-lock puzzle. All of the assumptions need to be sub-exponentially secure.*

Indistinguishability obfuscation is a method for "scrambling" programs in such a way that the implementation details are "hidden" in some (mathematically precise) sense. While this concept was suggested more than two decades ago by Barak et al. [BGI+01, BGI+12], it gained popularity only in the past 8 years after the works of Garg et al. [GGH+16] and Sahai and Waters [SW21] who paved the way for a remarkable variety of applications in cryptography and complexity theory. Very recently the existence of such an obfuscator was established either based on well-founded assumptions [JLS21a, JLS21b] (i.e., LPN, SXDH, and PRFs in NC^0, all sub-exponentially hard) or based on new yet concrete assumptions related to LWE [BDGM20, GP21, WW21, DQV+21].

Time-lock puzzles are, roughly speaking, puzzles that can be generated very efficiently and solved by running a highly sequential procedure. The security guarantee of such a puzzle is that any algorithm, even ones that use significant parallel processing power, cannot solve it much faster than the naive sequential procedure. They were proposed by Rivest, Shamir, and Wagner [RSW96] more than 25 years ago, and have since been extensively used in cryptography (for example, [BN00, Pin03, GMPY11, LPS20, KLX20, BDD+20, RS20, EFKP20, DKP21, BDD+21]). This construction is quantum broken since it relies on a hidden order (RSA) group. Apart from the most well-known construction based on the repeated-squaring assumption (proposed in [RSW96]), there is also a construction (due to Bitansky et al. [BGJ+16]) from indistinguishability obfuscation together with the *worst-case* assumption that non-parallelizing languages exist (a natural generalization of the assumption that $\mathsf{P} \not\subseteq \mathsf{NC}$).

Implications for (Classical) MPC. We note that it is possible to view our result via several different lenses in terms of classical MPC:

- Our MrNISC implies the first 2-round maliciously secure SPS MPC based on **well-founded falsifiable assumptions**. The only such previous known MrNISC was based on strong and non-standard assumptions [ABG+21].
- Our MrNISC implies the first 2-round maliciously secure SPS MPC with a **short and reusable first message**, based on any assumption. Namely, the first round message is not only independent of the function to be computed (which is necessary for reusability), but it is actually generated independently of the number of participating parties. All prior MPC protocols with this property only satisfy semi-malicious security in the plain model and require a trusted CRS to get malicious security [BGMM20, BL20, AJJM21, BGSZ21, BJKL21].
- Our MrNISC implies the first **concurrent** two-round maliciously secure SPS MPC. Indeed, at any point in time parties can join the protocol by publishing their input encodings, andeven start evaluation phases. This could happen even after some of the other parties have already published their input encodings and even participated in several evaluation phases. The only previously known *malicious* (SPS) concurrent MPC required three rounds [BGJ+17].

2 Technical Overview

In this section we give an overview of our constructions and the main ideas that are needed to prove its security.

2.1 Definition of Maliciously Secure MrNISC

Let us start by reviewing the syntax of MrNISC, as defined by Benhamouda and Lin [BL20].

Model and syntax. An MrNISC is a general purpose secure computation protocol with minimal communication pattern and maximum flexibility. Specifically, it consists of an input encoding phase which is done without any coordination with other parties in the system (i.e., without even knowing they exist), and an evaluation phase in which only relevant parties participate by publishing exactly one message each. In other words, MrNISC is a strict generalization of 2-round MPC with the following properties:

- there is no bound on the number of parties;
- multiple evaluation phases can take place with the same input encodings;
- parties can join at any point in time and publish their input encoding, even after multiple evaluation phases occurred.

We assume all parties have access to a broadcast channel which parties use to transmit message to all other parties. The formal syntax of an MrNISC consists of three polynomial-time algorithms $(\mathsf{Encode}, \mathsf{Eval}, \mathsf{Output})$, where Encode and Eval are probabilistic, and Output is deterministic. The allowed operations for a party P_i are:

- **Input Encoding phase**: each party P_i computes $m_{i,1}, \sigma_{i,1} \leftarrow \mathsf{Encode}(1^\lambda, x_i)$, where x_i is P_i's private input, $m_{i,1}$ is P_i's round 1 message, and $\sigma_{i,1}$ is P_i's round 1 private state. It broadcasts $m_{i,1}$ to all other parties.

- **Function Evaluation phase**: any set of parties I can compute an arity-$|I|$ function f on their respective inputs as follows. Each party P_i for $i \in I$ computes $m_{i,2} \leftarrow \mathsf{Eval}(f, \sigma_{i,1}, I, \{m_{j,1}\}_{j\in I})$, where f is the function to compute, x_i is P_i's private input, $\sigma_{i,1}$ is the private state of P_i's input encoding, $\{m_{j,1}\}_{j\in I}$ are the input encodings of all parties in I, and the output $m_{i,2}$ is P_i's round 2 message. It broadcasts $m_{i,2}$ to all parties in I
- **Output phase**: upon completion of the evaluation phase by each of the participating parties, anyone can compute $y \leftarrow \mathsf{Output}(\{m_{i,1}, m_{i,2}\}_{i\in I})$ which should be equal to $f(\{x_j\}_{j\in I})$.

Security. For security we require that an attacker does not learn any information beyond what is absolutely necessary, which is the outputs of the computations. Formally, for every "real-world" adversary that corrupts the evaluator and a subset of parties, we design an "ideal world" adversary (called a simulator) that can simulate the view of the real-world adversary using just the outputs of the computations. As in all previous works on MrNISC (including [BL20, AJJM21, BJKL21]), we assume static corruptions, namely that the adversary commits on the corrupted set of parties at the very beginning of the game. However, all previous works only achieved semi-malicious security (unless trusted setup assumptions are introduced). This notion of security, introduced by Asharov et al. [AJL+12], only considers corrupted parties that follow the protocol specification, except letting them choose their inputs and randomness arbitrarily. In contrast, we consider the much stronger and more standard notion of *malicious* security which allows the attacker to arbitrarily deviate from the specification of the protocol.

More precisely, in malicious security the adversary can behave arbitrarily in the name of the corrupted parties. Specifically, after the adversary commits on the corrupted set of parties, it can send an arbitrary round 1 message for a corrupted party, ask for a round 1 message of any honest party (with associated private input), ask an honest party to send the round 2 message corresponding to an evaluation of an arbitrary function on the round 1 message of an arbitrary set of parties, and send an arbitrary round 2 message of a malicious party corresponding to an evaluation of an arbitrary function on the round 1 message of an arbitrary set of parties. The simulator needs to simulate the adversary's view with the assistance of an ideal functionality that can provide only the outputs of the computations that are being performed throughout the adversary's interaction.

Typically, protocols are called *maliciously secure* if for every polynomial time adversary there is a polynomial time simulator for which the real-world experiment and the ideal-world experiment from above are indistinguishable. However, as mentioned, it is impossible to achieve such notion of malicious security for MPC (let alone MrNISC) in merely two rounds unless trusted setup assumptions are introduced. Therefore, we settle for super-polynomial simulation (SPS) which means that the simulator can run in super-polynomial time whereas the adversary is still assumed to run in polynomial time.

We refer to Section 4 for the precise definition.

Terminology. For the sake of brevity, we will sometimes refer to the *input encoding phase* as *round 1*, and the *function evaluation phase* as *round 2*.

2.2 The MrNISC Protocol

Recall that semi-malicious security only guarantees security when the adversary follows the honest protocol specification exactly (except that it can choose corrupted parties' randomness from arbitrary distributions). To achieve malicious security, we would like to use the following high-level approach, which is used by many classical MPC works.We require each party to commit to its input and

randomness as part of the input encoding phase, and then to prove using zero knowledge that all of its semi-malicious MrNISC messages were generated by following the prescribed protocol using that committed input and randomness. However, a problem arises when using this strategy with 2-round protocols. (Note that MrNISC requires that evaluation can be carried out in two rounds; in this way it is a strict generalization of 2-round MPC.) This problem comes from the fact that zero-knowledge in the plain model requires at least two rounds. Assuming we use such a 2-round ZK scheme, then honest parties would need to send their second-round MrNISC messages before finding out whether the first-round MrNISC messages were honest. This completely breaks security—if any party publishes semi-malicious messages based on a non-honest transcript, the semi-malicious protocol can make no security guarantees about these messages.

We need some way of overcoming this problem. That is, we need a way to publish second-round messages in such a way that they are only revealed if the first round is honest. To this end, we are going to attempt to use *witness encryption* as a locking mechanism: we "lock" the round 2 message of the underlying (semi-malicious) MrNISC and make sure that it can be unlocked only if all involved parties' proofs verify.

More precisely, party i does:

1. *Round 1 message*: Commit to its input and randomness and publish a round 1 message using the underlying MrNISC with the committed input/randomness pair. At the same time, generate a verifier's first-round ZK message for the other parties.

2. *Round 2 message*: Compute a round 2 message using the underlying MrNISC with randomness derived from the secret state. Generate a zero-knowledge proof that this was done correctly. Publish a witness encryption hiding the aforementioned round 2 message that could be recovered by supplying valid proofs that all other parties' first-round messages were created correctly.

With this template in mind, even before starting to think about how a security proof will look like, it is already evident that there are significant challenges is realizing the building blocks. Here are the three main challenges.

Challenge 1: The ZK argument system. The first challenge arises from trying to use ZK arguments as witnesses for the witness encryption scheme. Recall that witness encryption allows an encryptor to encrypt a message with respect to some statement Φ, and only if Φ is false then the message is hidden. Witness encryption (WE) crucially only can provide security when Φ is *false*; in particular, if Φ is true, even if it is computationally hard to find a witness for Φ, no guarantees are made about the encrypted message being hidden. Thus, it seems like we would need a *statistically-sound* ZK argument, i.e. a ZK proof: if the verifier's first round message is honest, with high probability there should not exist an accepting second-round ZK message.

It is well-known that to achieve ZK in two rounds, it is necessary to have a simulator which runs in super-polynomial time (i.e., a SPS simulator). In every such known two-round ZK, the simulator works by brute-forcing some trapdoor which was provided in round 1, and giving a proof that "either the statement is true or I found the trapdoor." Because of the existence of this trapdoor, it would be impossible to make any such ZK argument statistically sound: an unbounded-time machine can always find the trapdoor and prove false statements. So it seems like the ZK scheme needs to satisfy two contradictory requirements: be statistically sound, and be a two-round scheme (which seems to preclude statistical soundness).

Challenge 2: Non-malleability attacks. Since security of the underlying semi-malicious MrNISC holds only if the adversary knows some randomness for its messages, we need all parties to prove that they know the input and randomness corresponding to their messages. We are aiming for a protocol that can be evaluated in two rounds, so this necessitates using a non-malleable commitment (in order to prevent an attacker from say copying the round 1 message of some other party). Unfortunately, non-interactive non-malleable commitments without setup are only known from very strong non-standard assumptions, such as adaptive one-way functions [PPV08], hardness amplifiability [BL18, ABG+21], and/or keyless hash functions [BKP18, LPS20, BL18]. These are very strong and non-standard assumptions for some of which we have no plain-model instantiation, except heuristic ones. Thus, we want achieve a secure MrNISC protocol (in the plain model) without such strong assumptions.

Challenge 3: Adaptive reusability of the primitives. We emphasize that we are building a MrNISC protocol, which is a significant strengthening of standalone two-round MPC. Because of this, our ZK argument and commitment schemes must satisfy strong forms of reusability. There are several challenges in ensuring both the ZK argument and non-malleable commitment scheme satisfy the types of reusability that we need, and we introduce several new ideas in solving these challenges. We will elaborate on this challenge below, after we describe our ideas for solving challenges 1 and 2.

2.2.1 Solving Challenge 1: How do we get a "statistically-sound" SPS ZK?

We now discuss how to achieve the seemingly-contradictory requirements of getting a 2-round SPS ZK argument which has a statistical soundness property that would allow it to be a witness for the WE scheme. Our key idea is to relax the notion of statistical soundness to one that is obtainable in two rounds but still is sufficient to use with WE.

Imagine we have a WE scheme where the distinguishing advantage of an adversary is extremely small (say, subexponential in λ). It would then suffice to have a ZK protocol which is statistically sound a negligible fraction of the time, as long as it is quite a bit larger than the distinguishing advantage of the WE. We elaborate below.

Consider a hypothetical zero-knowledge protocol with the following properties:

- The first round between a polynomial time verifier and a prover fully specifies one of the two possible "modes": a *statistical ZK mode* and a *perfectly sound mode*.
- The perfectly sound mode occurs with some negligible probability ϵ, and in this mode no accepting round 2 message exists for any false statement
- In the statistical ZK mode (which occurs with overwhelming probability $1 - \epsilon$), the second message is simulatable by an SPS machine, and a simulated transcript is statistically indistinguishable from a normal transcript.
- Furthermore, it is computationally difficult for an adversarial prover to distinguish between the two modes.

If we had such a ZK protocol, it would enable us to argue hiding of the witness encryption scheme whenever the first round of the protocol is not honest. The idea of this argument is as follows. If an adversary could learn something about the second-round messages from their witness encryptions in some world where the first round was not honest, then it should also be able to do so even in the perfectly sound mode (otherwise it would distinguish the modes). But in this mode, proofs for false statements do not exist, and thus the witness encryption provides full security. Even though

this mode happens with negligible probability, it is still enough to contradict witness encryption security, whose advantage is much smaller.

To construct this new ZK scheme, we use ideas which are inspired by the extractable commitment scheme of Kalai, Khurana, and Sahai [KKS18]. This commitment scheme has the property that it is extractable with some negligible tunable probability, but is also statistically hiding. This commitment was used in the works of [BFJ+20] to get a two-round statistical zero knowledge argument with super-polynomial simulation. To instantiate our new "sometimes perfectly-sound" ZK argument, we use the protocol of [BFJ+20] as a starting point, but we will need to make significant modifications. Namely, in order to force a well-defined perfect soundness mode, we will make the first round of this protocol a "simultaneous-message" round, where both the prover and the verifier send a message. We elaborate further on this and other key ideas used in our construction in Section 2.2.1.

We note there is an important subtlety in this new definition and our construction. Namely, the statistical ZK and perfect soundness properties only hold with respect to the *second* round. If the verifier is unbounded-time, then after seeing a first-round prover's message it can send a first-round verifier's message that forces perfect soundness all the time, and thus disallows any prover from giving a simulated proof. On the other hand, if the prover is unbounded-time, then after seeing a first-round verifier's message it can send a first-round prover's message which causes the probability ϵ of the perfect soundness mode to be 0. Thus the frequency of perfect soundness mode and the ability of the simulator to give a simulated proof depend on the first round being generated by computationally bounded machines.

2.2.2 Solving Challenge 2: How do we avoid non-interactive non-malleability?

To solve challenge two, we must somehow get a non-malleable commitment (NMC) scheme which can be executed in the first round, without using strong assumptions such as keyless hash functions, hardness amlifiability, or adaptive one-way functions. Recall that unfortunately all known instantiations of non-interactive NMCs (for a super-polynomial number of tags) currently require the use of (some combination of) these strong assumptions, so it seems at first glance that avoiding them would require making substantial progress on the difficult and well-studied question of non-interactive NMCs.

Our approach to solving this problem is inspired by the exciting work of Khurana [Khu21], which builds a new type of commitment which works as follows. The commitment phase is similar to a non-interactive commitment in that the only communication from the committer is a first-round message C. The role of the receiver is slightly different: The receiver chooses a random string τ internally, and it is both C and τ together that truly defines the commitment (and, correspondingly, the underlying value being committed to). Consequently, in order to compute an opening, the committer must receive a τ from the receiver. Non-malleability (and binding) hinges upon the fact that the τ chosen by the receiver is chosen after seeing the commitment. (See the left diagram below for an illustration of this scheme.) Crucially, this commitment can be constructed from well-founded assumptions (indistinguishability obfuscation, time-lock puzzles, and OWPs), bypassing the need for the strong assumptions discussed earlier.

We would like to use this commitment scheme in our protocol. There are two main issues which arise.

- First, in order to use this scheme, we would need the commitment phase to happen entirely in the first round. Namely, the receiver must publish τ simultaneously while the committer is

Figure 1: The diagram on the left depicts the communication pattern of Khurana's [Khu21] commitment scheme, whereas the diagram on the right depicts ours. The key difference is that in our scheme, the receiver's message and the sender's messages can be sent simultaneously, while in [Khu21] the receiver's message must be sent after the sender's message.

publishing C. (See the right-hand diagram above.) In particular, in the security proof, we need to handle the case of malicious committers which publish C after seeing the round-1 τ.

- Second, our goal is to have every party use this commitment to commit to their input and randomness for the protocol. Recall that in the scheme of [Khu21], a well-defined commitment (C_j, τ_i) consists of *both* the committer's message C_j and the receiver's random string τ_i. Although honest parties P_j will always provide commitments C_j which are consistent across all τ_i, it is perfectly plausible for a corrupted party to publish some C_j where different τ_i yield commitments (C_j, τ_i) to different values.

Solving the first issue involves identifying some technical challenges in the security proof of [Khu21], and making changes to the protocol to avoid these issues. Roughly, we replace an encryption given in the first round with a time-lock puzzle-based commitment scheme. This allows us to carefully set the complexity hierarchy w.r.t. size and depth and thereby get security even if the τ's are chosen before C. For the second issue, by adding a standard (malleable) perfectly binding commitment (e.g., Blum's commitment) at the MrNISC protocol level, we are able to use this NMC scheme even though it does not satisfy the standard notion of binding.

We call this new primitive a *receiver-assisted one-round CCA-secure commitment*. We give more details and overview of the construction in Section 7.1.

2.2.3 Solving Challenge 3: How do we get reusability?

We now describe the challenges which arise when trying to get the type of reusability required by MrNISC. The main problem is to ensure that all of building blocks we use (i.e., the ZK scheme and the NMC scheme) support reuse of their first-round message. It turns out that the non-malleable commitment we described in the previous section can be adapted to this reusable setting without much modification. However, several challenges arise when trying to adapt the sometimes-statistically-sound ZK scheme which we discussed earlier to the reusable setting. We focus on these challenges here.

Recall that the ZK scheme is a simultaneous message protocol, so a transcript consists of three message of the form $(\mathsf{zk}_{1,P}, \mathsf{zk}_{1,V}, \mathsf{zk}_{2,P})$, a round-1 message of the prover and the verifier, and a round-2 message of the prover. What we need is for any prover to be able to publish a single $\mathsf{zk}_{1,P}$ in round 1 which can be used in many different sessions with respect to many different $\mathsf{zk}_{1,V}$ messages. In addition, we require a very strong form of reusability: even if a malicious verifier sees an entire transcript $(\mathsf{zk}_{1,P}, \mathsf{zk}_{1,V}, \mathsf{zk}_{2,P})$, and then chooses a new verifier's first-round message $\mathsf{zk}'_{1,V}$, zero knowledge should still hold when the prover publishes a proof with respect to $\mathsf{zk}'_{1,V}$ and the prover's *original* message $\mathsf{zk}_{1,P}$. In a similar manner, a verifier should be able to publish a single

$\mathsf{zk}_{1,V}$ which can be used in many different sessions with respect to many different $\mathsf{zk}_{1,P}$ messages, and the soundness properties of the ZK scheme should still hold.

Note that it is not immediately clear whether these reusability for ZK arguments are implied by corresponding non-reusable version of ZK arguments. In fact, this turns out not to be the case. In order to satisfy reusability, we end up having to make several changes to our (non-reusable) sometimes-perfectly-sound ZK scheme. We describe this in more detail in Section 9.

2.2.4 Putting things together

We now have the main pieces that we will use to construct a malicious-secure MrNISC: the two-round sometimes-statistically-sound ZK, receiver-assisted one-round CCA-secure commitment, and the underlying semi-malicious MrNISC. There are still significant challenges that arise when attempting to combine these pieces in the way described earlier to get a malicious MrNISC protocol. To see this, it will be convenient to briefly mention the approach we take for the security proof.

A simplified version of the sequence of hybrids we use is as follows. First, we extract the value underlying the commitments and check if anyone acted dishonestly. If so, we switch the honest parties' witness encryptions to encrypt 0 rather than the true round 2 message (this is hybrid 1). Second, we simulate the ZK proof (this is hybrid 2). Third, we switch the underlying value in the commitment to 0 (this is hybrid 3). Once the commitments are independent of the true input, we can use the simulator of the underlying MrNISC (this is hybrid 4). The last hybrid is identical to our simulator.

To make the transitions between the hybrids possible, we need to carefully set the hardness of every primitive. Each hybrid indistinguishability induces some hardness inequality for the involved primitives. Unfortunately, the inequalities seem to be in contradiction to each other. Observe that the first indistinguishability (between hybrid 0 and hybrid 1), we need that our ZK argument's soundness properties hold against adversaries who can run the CCA extractor. That is,

$$T_{\mathsf{extractor}} \ll T_{\mathsf{sound}}.$$

For the transition between hybrid 2 to 3, we need to guarantee that security of the commitment scheme holds even against an adversary that can run the ZK simulator. That is,

$$T_{\mathsf{ZKSim}} \ll T_{\mathsf{extractor}}.$$

Together, the above two inequalities imply that it is necessary to have $T_{\mathsf{ZKSim}} \ll T_{\mathsf{sound}}$. But this is impossible, at least using the techniques we use in constructing the ZK argument. Our simulator works by brute forcing the verifier's $\mathsf{zk}_{1,V}$ message to obtain some secret, and produces proofs with this knowledge. In other words, whoever has the secret can produce accepting proofs without knowing a witness—this is essentially an upper bound on the soundness of the scheme, i.e., $T_{\mathsf{sound}} \ll T_{\mathsf{ZKSim}}$, which means that our inequalities cannot be satisfied at the same time.

To solve this problem, we introduce another axis of hardness, namely, *circuit depth*. In particular, assume that it is possible to run the ZK simulator in some super-polynomial depth d. To do this, we would have to construct a ZK argument where the secret embedded in $\mathsf{zk}_{1,V}$ is extractable in depth d. Further, assume that in polynomial depth, it is extremely hard to extract the secret from $\mathsf{zk}_{1,V}$ (much harder than size d). We can use such a ZK argument to solve the problem above. Namely, we can restrict the reduction for hybrids 0 and 1 to run in *polynomial depth*, and in this complexity class it holds that $T_{\mathsf{extractor}} \ll T_{\mathsf{sound}}$. For the reduction for hybrids 2 and 3, we will allow the depth to be d, in which case the inequality $T_{\mathsf{ZKSim}} \ll T_{\mathsf{extractor}}$ is satisfied.

So we have reduced this problem to constructing a ZK argument which is simulateable in some super-polynomial depth d and whose soundness holds against size much larger than d as long as

the depth is restricted to be polynomial. It turns out that it is possible to modify our original ZK argument to satisfy this property; we describe this in Section 9, where we explain the ZK argument in detail.

There are several smaller technical issues that arise when putting things together. One such issue is that of "simulation soundness," that is, we need to guarantee that the adversary cannot give valid ZK arguments for false statements even if it sees simulated arguments from the honest parties. We solve this issue using techniques from the work of [BGJ+17]. The high-level idea is as follows. First, we ensure that a simulated ZK argument is indistinguishable from a normal ZK argument *even for an adversary that itself is powerful enough to run the simulator.* In our case, this means that even a

The high level idea is that if we use a ZK argument where the simulated proofs are indistinguishable from normal proofs even to an adversary which is powerful enough to run the simulator itself, and if we commit to the witnesses using a non-malleable commitment, it is possible to design a sequence of hybrids which guarantees simulation soundness.

This and other small technical details result in a construction and sequence of hybrids which is slightly more involved than the simplified version presented in this overview. We refer the reader to Section 6 for details.

3 Preliminaries

For any distribution $\mathcal{X}$, we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the distribution $\mathcal{X}$. For a set X we denote by $x \leftarrow X$ the process of sampling x from the uniform distribution over X. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, .., n\}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is negligible if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{negl}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$. Throughout, when we refer to polynomials in security parameter, we mean constant degree polynomials that take positive value on non-negative inputs. We denote by $\mathsf{poly}(\lambda)$ an arbitrary polynomial in λ satisfying the above requirements of non-negativity.

Throughout this paper, all machines are assumed to be non-uniform. We will use λ to denote the security. We will use PPT as an acronym for "probabilistic (non-uniform) polynomial-time". In addition, we use the notation $T_1 \ll T_2$ (or $T_2 \gg T_1$) if for all polynomials p, $p(T_1) < T_2$ asymptotically.

The statistical distance between two distributions X and Y over a discrete domain Ω is defined as $\Delta(X, Y) = (1/2) \cdot \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|$.

$(\mathcal{C}, \epsilon)$-indistinguishability. By $\mathcal{C}$ we denote an abstract class of adversaries, where each adversary $\mathcal{A} \in \mathcal{C}$ grows in some complexity measure (i.e. size, depth, etc) based on the security parameter λ. Security definitions will always hold with respect to some class of adversaries which we will specify.

Definition 1 ($(\mathcal{C}, \epsilon)$-Indistinguishability)**.** *Let $\epsilon : \mathbb{N} \to (0, 1)$ be a function. We say that two distribution ensembles $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ are $(\mathcal{C}, \epsilon)$-indistinguishable if for any adversary $\mathcal{A} \in \mathcal{C}$, for any polynomial* poly*, and any $\lambda \in \mathbb{N}$,*

$$\left| \Pr_{x \leftarrow \mathcal{X}_\lambda} \left[\mathcal{A}\left(1^\lambda, x\right) \right] - \Pr_{y \leftarrow \mathcal{Y}_\lambda} \left[\mathcal{A}\left(1^\lambda, y\right) \right] \right| \leq \epsilon(\lambda).$$

We use the shorthand $\mathcal{X} \approx_{(\mathcal{C}, \epsilon)} \mathcal{Y}$ to denote this. If $\mathcal{A}$ is unbounded time then we say that $\mathcal{Y}$ and $\mathcal{X}$ are statistically *indistinguishable and we write $\mathcal{X} \approx_{(\infty, \epsilon)} \mathcal{Y}$, or alternately $\Delta(\mathcal{X}, \mathcal{Y}) \leq \epsilon$. (This corresponds to the standard definition of statistical distance.)*

3.1 Witness Encryption

Here, we recall the definition of witness encryption, originally due to Garg et al. [GGSW13].

Definition 2. *A witness encryption scheme for an* NP *language L (with corresponding relation R) consists of the following two polynomial-time algorithms:*

- $\mathsf{WE.Enc}(1^\lambda, x, M)$*: The encryption algorithm takes as input the security parameter λ, a string $x \in \{0,1\}^*$, and a message $M \in \{0,1\}^*$. It outputs a ciphertext* CT*. This procedure is probabilistic.*
- $\mathsf{WE.Dec}(\mathsf{CT}, w)$*: The decryption algorithm takes as input a ciphertext* CT *along with a witness $w \in \{0,1\}^*$. It outputs a string $M \in \{0,1\}^*$ or the symbol $\perp$. This procedure is deterministic.*

These algorithms satisfy the following properties:

Correctness: *For any security parameter λ, for any message $M \in \{0,1\}^*$, any $x \in \{0,1\}^*$ such that $R(x,w) = 1$ for $w \in \{0,1\}^*$, we have that:*

$$\Pr[\mathsf{WE.Dec}(\mathsf{WE.Enc}(1^\lambda, x, M), w) = M] = 1.$$

$(\mathcal{C}, \epsilon)$-Security: *Fix any ensemble $\mathcal{X}_\lambda$ of polynomial length strings such that every $x \in \mathcal{X}_\lambda$ satisfies $x \notin L$, and any ensemble of messages $\mathcal{M}_\lambda$ of polynomial length. For every $\lambda \in \mathbb{N}, x \in \mathcal{X}_\lambda$, and $M \in \mathcal{M}_\lambda$, it holds that*

$$\mathsf{WE.Enc}(1^\lambda, x, M) \approx_{(\mathcal{C},\epsilon)} \mathsf{WE.Enc}(1^\lambda, x, 0^{|M|}).$$

It is well known that witness encryption can be obtained directly from indistinguishability obfuscation by ofuscating a circuit that has the instance x and the message M hardwired, gets as input a witness, and outputs M if the instance-witness pair verify.

Theorem 2. *Assuming a $(\mathcal{C}, \epsilon)$-indistinguishability obfuscator for all polynomial-size circuits, then there is a $(\mathcal{C}, \epsilon)$-witness encryption scheme for all* NP*.*

3.2 Indistinguishability Obfuscation

In this section, we define the notion of an indistinguishability Obfuscation.

Definition 3 (Indistinguishability Obfuscator (iO) for Circuits [BGI+01, BGI+12])**.** *A probabilistic polynomial-time algorithm* iO *is called a secure indistinguishability obfuscator for polynomial-sized circuits if the following holds:*

- ***Completeness****: For every $\lambda \in \mathbb{N}$, every circuit C with input length n, every input $x \in \{0,1\}^n$, we have that*
$$\Pr\left[\tilde{C}(x) = C(x) \ : \ \tilde{C} \leftarrow \mathsf{iO}(1^\lambda, C)\right] = 1 \ .$$
- **$(\mathcal{C}, \epsilon)$-*Indistinguishability:*** *For every two ensembles $\{C_{0,\lambda}\}_{\lambda\in\mathbb{Z}^+}$ and $\{C_{1,\lambda}\}_{\lambda\in\mathbb{Z}^+}$ of polynomial-sized circuits that have the same size, input length, and output length, and are functionally equivalent, that is, $\forall\lambda \in \mathbb{Z}^+$, $C_{0,\lambda}(x) = C_{1,\lambda}(x)$ for every input x, the distributions $\mathsf{iO}(1^\lambda, C_{0,\lambda})$ and $\mathsf{iO}(1^\lambda, C_{1,\lambda})$ are $(\mathcal{C}, \epsilon)$ indistinguishable.*

In this work, we require that iO is actually subexponentially secure against adversaries of subexponential size. As shown in [JLS21a, JLS21b] this can be instantiated assuming subexponential security of well studied hardness assumptions.

3.3 Correlation Intractable Hash Functions

We adapt definitions of a correlation intractable hash function family from [PS19, CCH+19].

Definition 4. *For any polynomials $k, (\cdot), s(\cdot) = \omega(k(\cdot))$ and any $\lambda \in \mathbb{N}$, let $\mathcal{F}_{\lambda,s(\lambda)}$ denote the class of NC^1 circuits of size $s(\lambda)$ that on input $k(\lambda)$ bits output λ bits. Namely, $f : \{0,1\}^{k(\lambda)} \to \{0,1\}^{\lambda}$ is in $\mathcal{F}_{\lambda,s}$ if it has size $s(\lambda)$ and depth bounded by $O(\log \lambda)$.*

We require the following property from such a function.

Definition 5 ($(\mathcal{C},\epsilon)$-Somewhere-Statistical Correlation Intractable Hash Function Family)**.** *A hash function family $\mathcal{H} = (\mathsf{FakeGen}, \mathsf{Eval})$ is $(\mathcal{C},\epsilon)$-somewhere-statistically correlation intractable (CI) with respect to $\mathcal{F} = \{\mathcal{F}_{\lambda,s(\lambda)}\}_{\lambda\in\mathbb{N}}$ as defined in Definition 4, if the following two properties hold:*

- ***Perfect Correlation Intractability:*** *For every $f \in \mathcal{F}_{\lambda,s}$ and every polynomial s,*

$$\Pr_{K\leftarrow\mathcal{H}.\mathsf{FakeGen}(1^\lambda,f)}\Big[\exists x \text{ such that } (x,\mathcal{H}.\mathsf{Eval}(K,x)) = (x,f(x))\Big] = 0.$$

- ***Computational Indistinguishability of Hash Keys:*** *Moreover, for every $f \in \mathcal{F}_{\lambda,s}$, for every $\mathcal{A} \in \mathcal{C}$, and every large enough $\lambda \in \mathbb{N}$,*

$$\Big|\Pr_{K\leftarrow\mathcal{H}.\mathsf{FakeGen}(1^\lambda,f)}[\mathcal{A}(K) = 1] - \Pr_{K\leftarrow\{0,1\}^\ell}[\mathcal{A}(K) = 1]\Big| < \epsilon(\lambda),$$

  where ℓ denotes the size of the output of $\mathcal{H}.\mathsf{Setup}(1^\lambda, f)$.

The work of [PS19] gives a construction of correlation intractable hash functions with respect to $\mathcal{F} = \{\mathcal{F}_{\lambda,s(\lambda)}\}_{\lambda\in\mathbb{N}}$, based on polynomial LWE with polynomial approximation factors. We observe that their construction also satisfies Definition 5, assuming LWE with an explicit efficiently computable advantage upper bound.

3.4 Time Lock Puzzles

We recall the notion of a time-lock puzzle scheme, originally due to [RSW96]. We adapt the definition from [BGJ+16].

Definition 6. *A D-secure time lock puzzle TLP is a tuple of two algorithms $(\mathsf{PGen}, \mathsf{Solve})$ that satisfies the following properties.*

Syntax:

- $\mathsf{PGen}(1^\lambda, 1^t, x)$: *The puzzle generation algorithm is a randomized polynomial time algorithm takes as input a security parameter λ and a hardness parameter t. It also takes as input a solution $x \in \{0,1\}^\lambda$. It outputs a puzzle Z.*
- $\mathsf{Solve}(Z)$ *The puzzle solving algorithm takes as input a puzzle Z. It outputs $x \in \perp \cup \{0,1\}^*$.*

Completeness: For every $\lambda, t \in \mathbb{N}$ and every $x \in \{0,1\}^\lambda$, $\Pr[\mathsf{Solve}(\mathsf{PGen}(1^\lambda, 1^t, x)) = x] = 1$.

Efficiency: PGen is a polynomial time algorithm in its input length, and $\mathsf{Solve}(Z)$ runs in time $\mathsf{poly}(2^t, \lambda)$ for every Z in support of $\mathsf{PGen}(1^\lambda, 1^t, \cdot)$.

D-security: Let $\lambda \in \mathbb{N}$, $t = t(\lambda) \in \lambda^{\Omega(1/\log\log\lambda)} \cap \lambda^{O(1)}$ and $x \in \{0,1\}^{\lambda^{\Theta(1)}}$. Then, it holds that for every Boolean circuit $\mathcal{A}$ with depth $D(t)$ and total size bounded by any polynomial in 2^λ it holds that:

$$\Big|\Pr[\mathcal{A}(\mathsf{PGen}(1^\lambda, 1^t, x)) = 1] - \Pr[\mathcal{A}(\mathsf{PGen}(1^\lambda, 1^t, 0^{|x|})) = 1]\Big| \le 2^{-\lambda}.$$

Note that we require security against sub-exponential size attackers and with sub-exponential distinguishing advantage. Specifically, we require that sub-exponential-size attackers (that are in depth at most $D(t)$) will not have advantage better than inverse sub-exponential. Sub-exponential *size* assumptions on the repeated squaring assumption were already made before, e.g., in [LPS20, DKP21, EFKP20]).

The first and most popular instantiation of time-lock puzzles was proposed by Rivest, Shamir, and Wagner [RSW96]. It is based on the "inherently sequential" nature of exponentiation modulo an RSA integer. That is, that t repeated squarings mod N, where $N = pq$ is a product of two secret primes, require "roughly" t depth. More than twenty years after their proposal, there still does not exist a (parallelizable) strategy that can solve such puzzles of difficulty parameter t in depth $D(t)$ which is significantly less than 2^t, with any non-trivial advantage. This is true even for the decision problem variant, rather than the search problem. (Note that the decision version is the one that is typically defined and assumed in constructions, e.g., [BN00, BGJ+16, LPS20, DKP21, EFKP20]).

Another construction of time-lock puzzles, due to Bitansky et al. [BGJ+16], based on indistinguishability obfuscation and (worst-case) non-parallelizing languages, is also an instantiation of the above definition, as long as the underlying are assumed to be sub-exponentially hard.

3.5 Sender Equivocal Oblivious Transfer

Definition 7 (Oblivious Transfer)**.** *An Sender-Equivocal Oblivious Transfer (OT) protocol consists of three randomized polynomial time algorithms:*

- $\mathsf{OT}_1(1^\lambda, b; r_1) \to \mathsf{ot}_1$: *The OT_1 algorithm takes as input a bit $b \in \{0,1\}$ and randomness r, and outputs the "receiver" message ot_1.*
- $\mathsf{OT}_2(\mathsf{ot}_1, m_0, m_1; r_2) \to \mathsf{ot}_2$: *The OT_2 algorithm takes as input a receiver message ot_1, two messages m_0, m_1, and randomness r_2, and it outputs the sender message ot_2.*
- $\mathsf{OT}_3(\mathsf{ot}_2, b, r_1) \to z$: *The OT_3 algorithm takes as input the sender message along with a bit $b \in \{0,1\}$ and randomness r_1. It outputs $z \in \perp \cup \{0,1\}^*$.*

We require a number of basic properties.
Correctness: Let $\lambda \in \mathbb{N}$, $b \in \{0,1\}$ and $(m_0, m_1) \in \{0,1\}^*$ with $|m_0| = |m_1|$. Then, it holds that:

$$\Pr[\mathsf{OT}_3(\mathsf{ot}_2, b, r_1) = m_b] = 1,$$

where $\mathsf{ot}_2 = \mathsf{OT}_2(\mathsf{ot}_1, m_0, m_1; r_2)$, $\mathsf{ot}_1 = \mathsf{OT}_1(1^\lambda, b; r_1)$ and probability is taken over the coins of r_1, r_2.

$(\mathcal{C}, \epsilon)$-Receiver Security: Let $\lambda \in \mathbb{N}$ be the security parameter. Then, it holds that:

$$\mathsf{OT}_1(1^\lambda, 0) \approx_{(\mathcal{C},\epsilon)} \mathsf{OT}_1(1^\lambda, 1).$$

Equivocation: There exist a polynomial time algorithm Equiv such that the following property is satisfied. For every $\lambda \in \mathbb{N}$ $b \in \{0,1\}$, $m_0, m_1 \in \{0,1\}^*$ with length ℓ, with probability 1 over the coins r_1 of $\mathsf{ot}_1 \leftarrow \mathsf{OT}_1(1^\lambda, b; r_1)$, the following two distributions are identically distributed. Let $v = (v_0, v_1)$ where $v_b = m_b$ and $v_{1-b} = 0^\ell$.

- *Distribution 1: Compute $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, m_0, m_1; r_2)$. Output $(b, r_1, \mathsf{ot}_2, m_0, m_1, r_2)$.*
- *Distribution 2: Compute $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(\mathsf{ot}_1, v_0, v_1; r_2')$ and $r_2 \leftarrow \mathsf{Equiv}(b, r_1, \mathsf{ot}_2, r_2', m_0, m_1)$. Output $(b, r_1, \mathsf{ot}_2, m_0, m_1, r_2)$.*

3.6 Equivocal Garbled Circuits for NC^1

Another primitive that we use is a an information theoretic variant of Yao's Garbled Circuits [Yao86a] for NC^1 circuits. This variant allows one to to efficiently "invert" the randomness used for garbling.

Definition 8 (Syntax)**.** *An information theoretic garbling scheme* $\mathsf{Gb} = (\mathsf{Garble}, \mathsf{Eval})$ *for circuit class* $\mathcal{F} = \{\mathcal{F}_\lambda\}_\lambda$ *(looking ahead, we will work with* $\mathsf{poly}(\lambda)$ *sized circuits with* λ *input bits, and depth* $O(\log \lambda)$*) consists of the following algorithm.*

- $\mathsf{Garble}(1^\lambda, C; r) \to (\Gamma, \{\mathsf{Lab}_{b,i}\}_{b\in\{0,1\}, i\in[\lambda]})$ *: The garbling algorithm takes as input a circuit* $C \in \mathcal{F}$*, and it outputs a garbled circuit* Γ *and input labels* $\{\mathsf{Lab}_{b,i}\}_{b\in\{0,1\},i\in[\lambda]}$*.*

  For any input $\mathbf{x}$*, we denote by* $\mathsf{Lab}_{\mathbf{x}}$ *the shorthand for* $\{\mathsf{Lab}_{x_i,i}\}_{i\in[\lambda]}$ *and* Lab *as the shorthand for* $\{\mathsf{Lab}_{b,i}\}_{b\in\{0,1\}}$*.*

- $\mathsf{Eval}(\Gamma, \{\mathsf{Lab}_{x_i,i}\}_{i\in[\lambda]}) \to z$ *: The evaluation algorithm takes as input a garbled circuit* Γ*, and labels* $\{\mathsf{Lab}_{x_i,i}\}_{i\in[\lambda]}$ *for some input* $\mathbf{x} \in \{0,1\}^\lambda$*. It outputs* $z \in \{0,1\}^* \cup \perp$*.*

We require that such a scheme satisfies the following properties:
Correctness: Let $\lambda \in \mathbb{N}$*,* $C \in \mathcal{F}$ *and* $\mathbf{x} \in \{0,1\}^\lambda$*, then it holds that:*

$$\Pr_{\mathsf{Garble}(1^\lambda, C)\to\Gamma, \{\mathsf{Lab}_{b,i}\}_{b\in\{0,1\},i\in[\lambda]}} [\mathsf{Eval}(\Gamma, \{\mathsf{Lab}_{x_i,i}\}_{i\in[\lambda]}) = C(\vec{x})] = 1$$

Equivocation: Let $\lambda \in \mathbb{N}$*,* $C_0, C_1 \in \mathcal{F}$ *and* $\mathbf{x} \in \{0,1\}^\lambda$ *such that* $C_0(\mathbf{x}) = C_1(\mathbf{x})$*, then the following two distributions are identical.*

- *Distribution* 1*: Compute* $(\Gamma, \mathsf{Lab}) \leftarrow \mathsf{Garble}(1^\lambda, C_1; r)$*. Output* $(C_1, \Gamma, \mathsf{Lab}, r)$*.*
- *Distribution* 2*: Compute* $(\Gamma, \mathsf{Lab}) \leftarrow \mathsf{Garble}(1^\lambda, C_0; r)$*. Compute* $\mathsf{GbEquiv}(\Gamma, \mathsf{Lab}_{\mathbf{x}}, C_1, \mathbf{x}) \to \mathsf{Lab}', r'$ *such that* $\mathsf{Lab}'_{x_i,i} = \mathsf{Lab}_{x_i,i}$ *for* $i \in [\lambda]$*. Output* $(C_1, \Gamma, \mathsf{Lab}', r')$*.*

Instantiation: To instantiate this, one can rely on the folklore instantiation of information-theoretic version of Yao's garbling scheme [Yao86b] for NC^1 circuits, and in particular the point-of-permute formulation of the scheme [Yao86b, BMR90].

4 MrNISC Syntax and Security

We define the syntax of MrNISC and formalize security notions for malicious adversaries as well as semi-malicious adversaries, following the general framework given by Benhamouda and Lin [BL20].

We assume all parties have access to a broadcast channel, which any party can use to transmit a message to all other parties. We consider protocols given in the form of three polynomial-time algorithms $(\mathsf{Encode}, \mathsf{Eval}, \mathsf{Output})$, where Encode and Eval are probabilistic and Output is deterministic, for which we define the syntax as follows:

- **Input Encoding phase**: each party P_i computes $m_{i,1} \leftarrow \mathsf{Encode}(1^\lambda, x_i; r_{i,1})$, where x_i is P_i's private input, and the output $m_{i,1}$ is P_i's round 1 message.
- **Function Evaluation phase**: any set of parties I can compute an arity-$|I|$ function f on their respective inputs as follows. Each party P_i for $i \in I$ computes $m_{i,2} \leftarrow \mathsf{Eval}(f, x_i, r_{i,1}, I, \{m_{i,1}\}_{i\in I}; r_{i,2})$, where f is the function to compute, x_i is P_i's private input, $r_{i,1}$ is the randomness which P_i used to generate its input encoding, $\{m_{i,1}\}_{i\in I}$ are the input encodings of all parties in I, and the output $m_{i,2}$ is P_i's round 2 message.
- **Output phase**: once the Eval phase ends, anyone can compute $y \leftarrow \mathsf{Output}(\{m_{i,1}, m_{i,2}\}_{i\in I})$.

Malicious security. We follow the standard real/ideal paradigm in the following definition. An MrNISC scheme is malicious-secure for every PPT adversary $\mathcal{A}$ in the real world there exists an ideal-world adversary $\mathcal{S}$ (the "simulator") such that the outputs of the following two experiments $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{S}}(\lambda)$ are indistinguishable.

Real experiment $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}(\lambda, z)$. The experiment initializes the adversary $\mathcal{A}$ with input auxiliary input z. In addition, the experiment initializes empty lists $\mathsf{functions\_learned}$ and $\mathsf{honest\_outputs}$. $\mathcal{A}$ chooses the number of parties M and the set of honest parties $H \subseteq [M]$. $\mathcal{A}$ then submits queries to the experiment in an arbitrary number of iterations until it terminates. In every iteration k, it can submit one query of one of the following four types.

- Corrupt Input Encoding: The adversary $\mathcal{A}$ can corrupt a party $i \notin H$ and send an arbitrary first message $m^*_{i,1}$ on its behalf.
- Honest Input Encoding: The adversary $\mathcal{A}$ can ask a party $i \in H$ to send its first message by running $m^*_{i,1} \leftarrow \mathsf{Encode}(1^\lambda, x_i; r_{i,1})$, where x_i is its input and $r_{i,1}$ is freshly chosen randomness.
- Honest Computation Encoding: The adversary $\mathcal{A}$ can ask an honest party $i \in H$ to evaluate a function f on the inputs of parties I. If all first messages of parties in I are already published, party i computes and publishes $m^*_{i,2} \leftarrow \mathsf{Eval}(f, x_i, I, r_{i,1}, \{m^*_{i,1}\}_{i\in I}; r_{i,2})$. Otherwise, the party outputs $\perp$. If $\mathcal{A}$ has received Eval messages with respect to f and I from all honest parties in I, the experiment adds the pair (f, I) to the list $\mathsf{functions\_learned}$.
- Corrupt Computation Encoding: The adversary can send an arbitrary function evaluation encoding to the honest parties on behalf of some corrupted party $i \notin H$ with respect to some function f and set I. If all parties in I have sent their Eval messages for (f, I), the experiment adds the honest parties' output $(f, I, \mathsf{Output}(\{m^*_{i,1}, m^*_{i,2}\}_{i\in I}))$ to the list $\mathsf{honest\_outputs}$.

The output of the real experiment is defined to be $(\mathsf{view}_{\mathcal{A}}, \mathsf{functions\_learned}, \mathsf{honest\_outputs})$, where $\mathsf{view}_{\mathcal{A}}$ is the view of $\mathcal{A}$ at the end of the computation, and $\mathsf{functions\_learned}$ and $\mathsf{honest\_outputs}$ are the two lists defined above.

Ideal experiment $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathcal{S}}(\lambda, z)$. The ideal experiment initializes $\mathcal{S}$ with auxiliary input z. In addition, the experiment initializes empty lists $\mathsf{functions\_learned}$ and $\mathsf{honest\_outputs}$. $\mathcal{S}$ chooses the number of parties M and the set of honest parties $H \subseteq [M]$. It then attempts to simulate the view of $\mathcal{A}$. To this end, it is allowed three types of queries to the ideal functionality $\mathcal{U}$:

- Declaring a corrupted party's input: $\mathcal{S}$ can send a pair (i, x^*_i) to $\mathcal{U}$, for $i \notin H$, which $\mathcal{U}$ uses as party i's input for all function evaluations.
- Learning a function evaluation: $\mathcal{S}$ can send a pair (f, I) to $\mathcal{U}$. If for all $i \in I \setminus H$ $\mathcal{S}$ has declared party i's input, $\mathcal{U}$ computes the evaluation y of f over the inputs of all parties in I, and returns it to $\mathcal{S}$. $\mathcal{U}$ then adds (f, I) to the list $\mathsf{functions\_learned}$.
- Delivering output to honest parties: For any $(f, I) \in \mathsf{functions\_learned}$, $\mathcal{S}$ can ask $\mathcal{U}$ to deliver the corresponding output y to the honest parties. In that case, $\mathcal{U}$ adds (f, I, y) to the list $\mathsf{honest\_outputs}$.

The output of the ideal experiment is defined to be $(\widehat{\mathsf{view}}, \mathsf{functions\_learned}, \mathsf{honest\_outputs})$, where $\widehat{\mathsf{view}}$ is the simulated view of $\mathcal{A}$ which is output by $\mathcal{S}$ at the end of the computation, and $\mathsf{functions\_learned}$ and $\mathsf{honest\_outputs}$ are the two lists defined above.

Definition 9 (($\mathcal{C}_{\mathsf{adv}}, \mathcal{C}_{\mathsf{sim}}, \epsilon$)-Maliciously Secure MPC)**.** *We say that an MPC protocol* Π *is* ($\mathcal{C}_{\mathsf{adv}}, \mathcal{C}_{\mathsf{sim}}, \epsilon$)-maliciously secure *if for every* $\mathcal{C}_{\mathsf{adv}}$ *adversary* $(\mathcal{A}, \mathcal{D})$ *there exists a* $\mathcal{C}_{\mathsf{sim}}$ *ideal-world adversary* $\mathcal{S}$ *(i.e., the simulator) such that for every string* z*,*

$$\left| \Pr\left[\mathcal{D}(\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}(\lambda, z)) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathcal{S}}(\lambda, z)) = 1\right] \right| < \epsilon(\lambda).$$

The standard notion of security requires for every polynomial $p(\cdot)$ the existence of a polynomial $q(\cdot)$ for which the protocol is (p, q, ϵ)-maliciously secure, where $\epsilon(\cdot)$ is a negligible function. However, since we are interested in two-round MPC protocols, it is known that the standard polynomial notion of security is impossible. Therefore, we focus on the relaxed notion of super-polynomial security (SPS): there is a sub-exponential function $q(\cdot)$ such that for all polynomials $p(\cdot)$, the protocol is (p, q, ϵ)-maliciously secure.

The semi-malicious case. We define a variant of the above security definition which closely mirrors the definition of semi-malicious secure multiparty computation [AJW11]. A *semi-malicious MrNISC adversary* is modeled as an algorithm which, whenever it sends a corrupted input or computation encoding on behalf of some party P_j, must also output some pair (x, r) which *explains its behavior*. More specifically, all of the protocol messages sent by the adversary on behalf of P_j up to that point, including the message just sent, must exactly match the honest protocol specification for P_j when executed with input x and randomness r. Note that the witnesses given in different rounds need not be consistent. We allow the adversary to send a special "abort" Corrupt Computation Encoding message, where it instructs all parties in some evaluation (f, I) to set their outputs to $\perp$. In this sense, the adversary may abort any individual function evaluation. The adversary may also choose to abort the entire execution in any step of the interaction.

Definition 10 (($\mathcal{C}_{\mathsf{adv}}, \mathcal{C}_{\mathsf{sim}}, \epsilon$)-Semi-Malicious Secure MPC)**.** *We say that an MPC protocol* Π *is* ($\mathcal{C}_{\mathsf{adv}}, \mathcal{C}_{\mathsf{sim}}, \epsilon$)-semi-malicious secure *if for every* $\mathcal{C}_{\mathsf{adv}}$ *semi-malicious adversary* $(\mathcal{A}, \mathcal{D})$ *there exists* $\mathcal{C}_{\mathsf{sim}}$ *ideal-world adversary* $\mathcal{S}$ *(i.e., the simulator) such that for every string* z*,*

$$\left| \Pr\left[\mathcal{D}(\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}(\lambda, z)) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{A},\mathcal{S}}(\lambda, z)) = 1\right] \right| < \epsilon(\lambda).$$

5 Main Building Blocks

5.1 Reusable Statistical ZK Arguments with Sometimes-Statistical Soundness

We define statistical zero-knowledge arguments with a specific communication pattern. The particular protocol that we need has a "simultaneous message" first round, where both prover and verifier will send a message at the same time. The syntax is the following:

1. The (honest) prover $P = (\mathsf{ZKProve}_1, \mathsf{ZKProve}_2)$ and verifier $V = (\mathsf{ZKVerify}_1, \mathsf{ZKVerify}_2)$ are each composed of two uniform PPT algorithms.

2. $\mathsf{ZKProve}_1$ and $\mathsf{ZKVerify}_1$ get as input only the security parameter λ. $\mathsf{ZKProve}_1$ outputs a message $\mathsf{zk}_{1,P}$ and a state σ_P. $\mathsf{ZKVerify}_1$ outputs a message $\mathsf{zk}_{1,V}$ and a state σ_V. The first round transcript is denoted $\tau_1 = (\mathsf{zk}_{1,P}, \mathsf{zk}_{1,V})$.

3. $\mathsf{ZKProve}_2$ gets σ_P, $\mathsf{zk}_{1,V}$, the instance x, and a witness w. It outputs a message $\mathsf{zk}_{2,P}$.

4. $\mathsf{ZKVerify}_2$ gets the instance x and $\tau = (\tau_1, \mathsf{zk}_{2,P})$, and outputs $0/1$.

Looking ahead, we shall consider two-round ZK protocols as above with super-polynomial simulation (SPS), i.e., the simulator can run longer than the soundness bound. Further, we will also require that for a given prover and a verifier, the first message is reusable for proving multiple statements. We denote $\langle P(w), V\rangle(1^\lambda, x)$ the output of the interaction between P and V, where P gets as input the witness w, and both P and V receive the instance x as a common input.

Definition 11 (Statistical Zero-Knowledge Arguments)**.** *Let L be a language in NP with a polynomial-time computable relation R_L. A protocol between P and V is a $(\mathcal{C}_{\mathsf{sound}}, \mathcal{C}_{\mathcal{S}}, \mathcal{C}_{\mathsf{zk}}, \epsilon_{\mathcal{S}}, \epsilon_{\mathsf{sound}})$-statistical zero-knowledge argument for L if it satisfies the following properties:*

- ***Perfect Completeness.*** *For every security parameter 1^λ and $(x, w) \in R_L$, it holds that*

$$\Pr\left[\langle P(w), V\rangle(1^\lambda, x) = 1\right] = 1,$$

  where the probability is over the random coins of P and V.

- ***$(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound}})$-Soundness.*** *For every polynomial $p(\lambda)$ and every prover $P^* \in \mathcal{C}_{\mathsf{sound}}$ that given 1^λ, chooses an input length 1^p for some polynomial $p \in \mathsf{poly}(\lambda)$, and then chooses $x \in \{0,1\}^p \setminus L$, it holds that*

$$\Pr\left[\langle P^*, V\rangle(1^\lambda, x) = 1\right] \leq \epsilon_{\mathsf{sound}}(\lambda),$$

  where the probability is over the random coins of V.

- ***$(\mathcal{C}_{\mathcal{S}}, \mathcal{C}_{\mathsf{zk}}, \epsilon_{\mathcal{S},1}, \epsilon_{\mathcal{S},2})$-Statistical Zero-Knowledge.*** *There exists a (uniform) simulator $\mathcal{S} \in \mathcal{C}_{\mathcal{S}}$ which takes as input the round-one transcript τ_1, the honest prover's state σ_P, and a statement x such that the following holds. Consider an adversary $V^* \in \mathcal{C}_{\mathsf{zk}}$ that takes as input 1^λ and an honestly generated prover's first round message $\mathsf{zk}_{1,P}$ and outputs a verifier's first round message $\mathsf{zk}^*_{1,V}$. Then, for all $(x, w) \in R_L$,*

$$\Pr\left[\Delta(\mathcal{S}(\sigma_p, \tau_1, x), \mathsf{ZKProve}_2(\sigma_p, \tau_1, x, w)) \geq \epsilon_{\mathcal{S},1}\right] < \epsilon_{\mathcal{S},2},$$

  where $\tau_1 = (\mathsf{zk}_{1,P}, \mathsf{zk}^*_{1,V})$ and $\Delta(X, Y)$ is the statistical distance between two distributions X and Y.

  In particular, if we define $\mathsf{expt}^0_{V^*,\mathsf{zk}}$ to be the output of the experiment above where the simulator is used to generate $\mathsf{zk}_{2,P}$, and $\mathsf{expt}^1_{V^*,\mathsf{zk}}$ such that the honest prover is used instead, this implies that

$$\mathsf{expt}^0_{V^*,\mathsf{zk}} \approx_{(\infty, \epsilon_{\mathcal{S},1}+\epsilon_{\mathcal{S},2})} \mathsf{expt}^1_{V^*,\mathsf{zk}}.$$

Additionally, we need a refined soundness property, defined next.

Definition 12 ($(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$-statistical soundness)**.** *Consider any prover $P^* \in \mathcal{C}_{\mathsf{sound}}$ and a polynomial $p(\cdot)$, where on input the security parameter 1^λ, P^* outputs an instance $x \in \{0,1\}^p \setminus L$. We require that there exists a "soundness mode indicator" machine $\mathcal{E}$ that on input $(\tau_1, \mathsf{state}_V)$ outputs either 0 or 1 such that the following properties hold.*

- ***Frequency of Soundness Mode.*** *For every prover $P^* \in \mathcal{C}_{\mathsf{sound}}$,*

$$\Pr\left[\mathcal{E}(\tau_1, \mathsf{state}_V) = 1\right] \geq \epsilon_{\mathsf{sound},1}(\lambda),$$

  where the probability is over the coins of the prover and the verifier in round 1.

- ***Perfect Soundness Holds During Soundness Mode.*** *For every prover $P^* \in \mathcal{C}_{\mathsf{sound}}$ and every round-1 state $(\tau_1, \mathsf{state}_{P^*}, \mathsf{state}_V)$ of the protocol, if $\mathcal{E}(\tau_1, \mathsf{state}_V) = 1$ then for all second-round messages $\mathsf{zk}_{2,P}$ sent by the prover corresponding to some false statement $x \notin L$, the verifier rejects on input $(x, \tau_1, \mathsf{zk}_{2,P}, \mathsf{state}_V)$.*

- ***Indistinguishability of Soundness Mode.*** *For every prover $P^* \in \mathcal{C}_{\mathsf{sound}}$, it holds that*

$$\{(\tau_1, \mathsf{state}_{P^*}) \mid \mathcal{E}(\tau_1, \mathsf{state}_V) = 1\} \approx_{(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},2})} \{(\tau_1, \mathsf{state}_{P^*}) \mid \mathcal{E}(\tau_1, \mathsf{state}_V) = 0\}.$$

The full MrNISC protocol needs a powerful version of zero knowledge, as follows:

Definition 13 ($(\mathcal{C}_{\mathcal{S}}, \mathcal{C}_{\mathsf{zk}}, \epsilon_{\mathcal{S}})$-Adaptive Reusable Statistical Zero-Knowledge)**.** *We say a zero knowledge scheme satisfies $(\mathcal{C}_{\mathcal{S}}, \mathcal{C}_{\mathsf{zk}}, \epsilon_{\mathcal{S},1}, \epsilon_{\mathcal{S},2})$-adaptive reusable statistical zero-Knowledge if there exists a (uniform) simulator $\mathcal{S} \in \mathcal{C}_{\mathcal{S}}$ which takes as input the round-one transcript τ_1, the honest prover's state σ_P, and a statement x such that the following holds. Consider an adversary $V^* \in \mathcal{C}_{\mathsf{zk}}$ that takes as input 1^λ and an honestly generated prover's first round message $\mathsf{zk}_{1,P}$, and plays the following game $\mathsf{expt}^b_{V^*,\mathsf{zk}}$:*

1. *V^* may adaptively issue queries of the form $(x, w, \mathsf{zk}^*_{1,V})$. The challenger responds as follows:*
   - *$f\ (x, w) \notin R_L$, the challenger responds with $\perp$.*
   - *If $(x, w) \in R_L$ and $b = 0$, the challenger responds with the honest prover's second message $\mathsf{ZKProve}_2(\sigma_p, \mathsf{zk}^*_{1,V}, x, w)$.*
   - *If $(x, w) \in R_L$ and $b = 1$, the challenger responds with the simulated prover's message $\mathcal{S}(\sigma_p, \mathsf{zk}^*_{1,V}, x)$.*
2. *At the end of the game, V^* outputs an arbitrary function of its view, which is used as the output of the experiment.*

It must hold that

$$\mathsf{expt}^0_{V^*,\mathsf{zk}} \approx_{(\infty, \epsilon_{\mathcal{S}})} \mathsf{expt}^1_{V^*,\mathsf{zk}}.$$

An overview and full details of our construction of the reusable SZK argument with sometimes-statistical soundness can be found in Section 9.

5.2 Receiver-Assisted One-Round CCA-Secure Commitments

We define the notion of receiver-assisted one-round CCA-secure commitments, denoted by aCCA. This section will largely follow the terminology Khurana [Khu21]. Let $\mathcal{T} = \{\mathcal{T}_\lambda\}_{\lambda \in \mathbb{N}}$ be the tag space which is $[T(\lambda)]$, where $T = 2^{\mathsf{poly}(\lambda)}$. Let the message space be $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$. We will have $\mathcal{M}$ be $\{0,1\}^{\ell_m(\lambda)}$ for some polynomial $\ell_m(\cdot)$.

Definition 14 (Syntax of aCCA)**.** *A receiver-assisted one-round CCA-secure commitment scheme aCCA for the message space $\mathcal{M}$, and tag space $\mathcal{T}$, consists of the following algorithms.*

$\mathsf{CCACommit}(1^\lambda, \mathsf{tag}, m; r)$: *The probabilistic polynomial time commitment algorithm takes as input the security parameter λ, a tag $\mathsf{tag} \in \mathcal{T}_\lambda$, a message $m \in \mathcal{M}_\lambda$ and it outputs a commitment P.*

$\mathsf{Opening}(\tau, \mathsf{tag}, \mathsf{P}, m, r)$: *The polynomial time deterministic algorithm Opening takes as input a string $\tau \in \{0,1\}^{\ell_t}$, a tag $\mathsf{tag} \in \mathcal{T}_\lambda$, a commitment P, a message $m \in \mathcal{M}_\lambda$, and randomness r. It outputs an opening $\sigma \in \{0,1\}^*$. Above $\ell_t = \ell_t(\lambda, n)$ is a polynomial associated with the scheme.*

$\mathsf{Open}(\tau, \mathsf{tag}, \mathsf{P}, m, \sigma)$: *The polynomial time deterministic algorithm* Open *takes a string* $\tau \in \{0,1\}^{\ell_t}$*, a tag* $\mathsf{tag} \in \mathcal{T}_\lambda$*, a commitment* P*, a message* $m \in \mathcal{M}_\lambda$*, and an opening* σ*. It outputs a value in* $\{0,1\}$*.*

Such a scheme satisfies following properties:

Definition 15 (Correctness of Opening)**.** *Let* $\lambda \in \mathbb{N}$ *be the security parameter, and consider any* $\mathsf{tag} \in \mathcal{T}_\lambda$*, any message* $m \in \mathcal{M}_\lambda$*, any* $\tau \in \{0,1\}^{\ell_t}$*, any* $\mathsf{P} \leftarrow \mathsf{CCACommit}(1^\lambda, \mathsf{tag}, m; r)$*. Then,*

$$\Pr[\mathsf{Open}(\tau, \mathsf{tag}, \mathsf{P}, m, \sigma) = 1] = 1,$$

where $\sigma = \mathsf{Opening}(\tau, \mathsf{tag}, \mathsf{P}, m, r)$

Definition 16 (Extraction)**.** *There exists an (inefficient) algorithm* CCAVal *with the following properties. For any* $\lambda \in \mathbb{N}$ *and any message* $m \in \mathcal{M}_\lambda$*, tag* $\mathsf{tag} \in \mathcal{T}_\lambda$*, commitment* P*, and* $\tau \in \{0,1\}^{\ell_t(\lambda)}$*, it holds that*

$$\exists \sigma'; \mathsf{Open}(\tau, \mathsf{tag}, \mathsf{P}, m, \sigma') = 1 \iff \mathsf{CCAVal}(\tau, \mathsf{tag}, \mathsf{P}) = m = 1.$$

In addition, CCAVal *runs in time* $2^{\mathsf{poly}(\lambda)}$ *for some fixed polynomial* poly*.*

We now define the CCA security property associated with the scheme:

Definition 17 ($(\mathcal{C}, \epsilon)$-CCA security)**.** *We define the following security game played between the adversary* $\mathcal{A} \in \mathcal{C}$ *and the challenger. We denote it by* $\mathsf{expt}_{\mathcal{A},\mathsf{CCA}}(1^\lambda)$*:*

1. *The challenger manages a list* L *that is initially empty. The contents of the list are visible to the adversary at all stages.*
2. *The adversary sends a challenge tag* $\mathsf{tag}^* \in \mathcal{T}_\lambda$*.*
3. *The adversary submits queries of the following kind in an adaptive manner:*
   (a) *Adversary can ask for arbitrary polynomially many* τ*-query. Challenger samples* $\tau' \leftarrow \{0,1\}^{\ell_t}$ *and appends* τ' *to* L*.*
   (b) *Adversary can ask for an arbitrary polynomially many* $(\tau, \mathsf{tag}, \mathsf{P})$*-query for any* $\tau \in L$*, any* $\mathsf{tag} \neq \mathsf{tag}^*$*, and any commitment* P*. The challenger computes* $\mathsf{CCAVal}(\tau, \mathsf{tag}, \mathsf{P})$ *and sends the result to the adversary.*
4. *The adversary submits two messages* $m_0, m_1 \in \mathcal{M}_\lambda$*. The challenger samples* $b \leftarrow \{0,1\}$*, and computes* $\mathsf{P}^* \leftarrow \mathsf{CCACommit}(1^\lambda, \mathsf{tag}^*, m_b)$*. The adversary gets* P^* *from the challenger.*
5. *The adversary repeats Step 3.*
6. *Finally, the adversary outputs a guess* $b' \in \{0,1\}$*. The experiment outputs* 1 *if* $b' = b$ *and* 0 *otherwise.*

The receiver-assisted one-round CCA*-secure commitment scheme* aCCA *scheme satisfies* $(\mathcal{C}, \epsilon)$-CCA *security if for all adversaries* $\mathcal{A} \in \mathcal{C}$*:*

$$\left| \Pr[\mathsf{expt}_{\mathcal{A},\mathsf{CCA}}(1^\lambda) = 1] - \frac{1}{2} \right| \leq \epsilon.$$

Remark 1. *The construction in [Khu21], although not specified exactly using these properties, satisfies a weaker notion of security than the one definition above. In this weaker notion, the adversary is restricted to output all commitments C_i for which it asks a query of the form* $\mathsf{CCAVal}(\star, \star, C_i)$ before *asking τ/ oracle queries. Our construction, which is a minor modification of [Khu21]'s satisfy the stronger notion defined above.*

Remark 2. *We also consider a weaker notion of security where the adversary queries* CCAVal *oracle at at most one* $\mathsf{tag} \neq \mathsf{tag}^*$. *We call this as the CCA security with one-tag restriction. There are general method to convert a scheme satisfying CCA security with the one tag restriction to one which satisfies the stronger security definition above (e.g., [Khu21]).*

Our construction of a receiver-assisted CCA secure commitment is largely based on ideas of Khurana [Khu21]. We make few crucial modifications to their construction in order to satisfy the above definition. We refer to Section 7 for an overview and full details.

6 Malicious-Secure MrNISC

In this section, we prove Theorem 1.

Required Primitives and Parameters. We make use of the following primitives in our construction.

- Commitment: A non-interactive perfectly binding commitment ($\mathsf{NICommit}$).
- Pseudo-Random Function A pseudo-random function (PRF).
- Witness Encryption: We use witness encryption as in Definition 2. We use circuit SAT as our NP language.
- statistical ZK arguments with SPS simulation: We use the SPS ZK argument ($\mathsf{ZKProve}_1, \mathsf{ZKVerify}_1, \mathsf{Z}$ satisfying Definitions 11 to 13 for circuit SAT constructed in Section 9.
- Receiver-assisted CCA commitments: We use receiver-assisted CCA commitments as in Definitions 14 to 17.
- Semi-malicious MrNISC: We use an underlying semi-malicious MrNISC protocol ($\mathsf{SM.Encode}, \mathsf{SM.Eval}, \mathsf{SM.}$ satisfying the security notion given in Definition 10. We assume that the simulator has the following syntax:
  - $\mathsf{SMSim}_1(1^\lambda, \mathcal{C}) \to (\{\hat{m}_{i,1}\}_{i\notin\mathcal{C}}, \sigma_{\mathcal{S}})$ takes as input the set $\mathcal{C}$ of corrupted parties, and outputs a list $\{\hat{m}_{i,1}\}_{i\notin\mathcal{C}}$ of simulated honest parties' input encodings along with a simulator's state $\sigma_{\mathcal{S}}$.
  - $\mathsf{SMSim}_2(\sigma_{\mathcal{S}}, I, f, \{\tilde{x}_j, \tilde{r}_j, \hat{m}_{j,1}\}_{j\in\mathcal{C}}, y) \to \{\hat{m}_{i,2}\}_{i\in I\setminus\mathcal{C}}$ takes as input a simulator's state $\sigma_{\mathcal{S}}$ along with a function f and a set I of parties who are participating in the evaluation of f, a list $\{\tilde{x}_j, \tilde{r}_j, \hat{m}_{j,1}\}_{j\in\mathcal{C}}$ of each corrupted party P_j's input, the randomness P_j used to generate its input encoding, and the input encoding itself, and a function evaluation y, and outputs a list $\{\hat{m}_{i,2}\}_{i\in I\setminus\mathcal{C}}$ of simulated honest parties' function evaluation encodings.

- **Complexity hierarchy.**

  In order to argue security, we require that the primitives we use are secure against adversaries of varying complexities. In particular, we require the following complexity hierarchy to hold with respect to the primitives. Let T_1, T_2, T_3, T_4, T_5 be functions over λ, such that

  $$T_1 \ll T_2 \ll T_3 \ll T_4 \ll T_5,$$

  where $T \ll T'$ means that $p(T) < T'$ asymptotically for all polynomials p. We require the following:

  - The ZK argument scheme satisfies $(\mathcal{C}_{\mathcal{S}}, \mathcal{C}_{\mathsf{zk}}, \epsilon_{\mathcal{S}})$-adaptive reusable statistical zero knowledge (Definition 13) where $\mathcal{C}_{\mathcal{S}}$ is the class of circuits of size $\mathsf{poly}(T_1)$ and depth T_1 (i.e. the simulator runs in size $\mathsf{poly}(T_1)$ and depth T_1), and $\mathcal{C}_{\mathsf{zk}}$ is the class of circuits of size $p(T_3)$ for all polynomials p, and $\epsilon_{\mathcal{S}}$ is any negligible function (i.e. statistical zero knowledge holds as long as the verifier's first-round message is generated by a machine in $\mathcal{C}_{\mathsf{zk}}$.
  - The CCA non-malleable commitment scheme satisfies $(\mathcal{C}, \epsilon)$-CCA security, where $\mathcal{C}$ is the class of circuits of size $p(T_1)$ for all polynomials p, and ϵ is any negligible function.
  - The CCA non-malleable commitment scheme's extractor CCAVal is a circuit of size T_2 and polynomial depth.
  - The perfectly-binding commitment scheme is hiding against adversaries of size $p(T_2)$ for all polynomials p, and is extractable by a circuit of size T_3.
  - The ZK argument scheme satisfies $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$-statistical soundness, where $\mathcal{C}_{\mathsf{sound}}$ is the class of circuits of size $p(T_5)$ and polynomial depth for all polynomials p (refer to Definition 12 for details on the meaning of $\mathcal{C}_{\mathsf{sound}}$), and $\epsilon_{\mathsf{sound},1} = 1/T_4$, and $\epsilon_{\mathsf{sound},2}$ is any negligible function.
  - The witness encryption scheme satisfies $(\mathcal{C}, \epsilon)$-security, where $\mathcal{C}$ is the class of circuits of size $p(T_5)$ for all polynomials p, and $\epsilon = 1/T_5$.
  - The pseudo-random function is secure against adversaries of size $p(T_5)$ for all polynomials p.
  - The semi-malicious MrNISC protocol is secure against adversaries of size $p(T_5)$ for all polynomials p.

Protocol. We give the protocol below, described in terms of the behavior of party P_i during the input encoding phase, the evaluation phase, and the output computation phase. In particular, we give this behavior by implementing the Encode, Eval and Output algorithms defined in Section 4. Assume that each party P_i has input x_i and a public identity denoted by $\mathsf{tag}_i \in \mathcal{T}_\lambda$. Note that the Output algorithm is public and can be performed without P_i's private input or state. Throughout the protocol description, we deal with PPT algorithms as follows. If a PPT algorithm P is invoked on some input x without any randomness explicitly given (i.e., we write $P(x)$), we implicitly assume that it is supplied with freshly chosen randomness. In some cases we will need to explicitly manipulate the randomness of algorithms, in which case we will write $P(x; r)$.

- **Input Encoding** $\mathsf{Encode}(1^\lambda, \mathsf{tag}_i, x_i)$**:** The input encoding algorithm takes as input 1^λ, where λ is the security parameter, along with P_i's tag tag_i and private input x_i, and does the following.

1. Compute the semi-malicious input encoding $\hat{m}_{i,1} \leftarrow \mathsf{SM.Encode}(1^\lambda, x_i; r_{i,\mathsf{SM},1})$, where $r_{i,\mathsf{SM},1} \xleftarrow{\$} \{0,1\}^*$ is freshly chosen randomness.
2. Choose a PRF key K_i.
3. Compute a perfectly binding commitment $\mathsf{com}_i \leftarrow \mathsf{NICommit}(1^\lambda, (x_i, r_{i,\mathsf{SM},1}, K_i); r_{i,\mathsf{com}})$ of the input and the semi-malicious encoding randomness, where $r_{i,\mathsf{com}} \xleftarrow{\$} \{0,1\}^*$ is freshly chosen randomness.
4. Compute a CCA-non-malleable commitment

$$\mathsf{nmc}_i \leftarrow \mathsf{CCACommit}(1^\lambda, \mathsf{tag}_i, (x_i, r_{i,\mathsf{SM}}, K_i, r_{i,\mathsf{com}}); r_{i,\mathsf{CCA}})$$

of the same values committed to in the perfectly binding commitment, along with the randomness used for generating the perfectly binding commitment, where $r_{i,\mathsf{CCA}} \xleftarrow{\$} \{0,1\}^*$ is freshly chosen randomness.
5. Compute a random string $\tau_i \xleftarrow{\$} \{0,1\}^\ell$.
6. Compute the first round verifier's message and state $(\sigma_{\mathsf{zk},1,i,V}, \mathsf{zk}_{1,i,V}) \leftarrow \mathsf{ZKVerify}_1(1^\lambda)$ and the first round prover message and state $(\sigma_{\mathsf{zk},1,i,P}, \mathsf{zk}_{1,i,P}) \leftarrow \mathsf{ZKProve}_1(1^\lambda)$.
7. Output $m_{i,1} = (\hat{m}_{i,1}, \mathsf{com}_i, \mathsf{nmc}_i, \tau_i, \mathsf{zk}_{1,i,V}, \mathsf{zk}_{1,i,P})$.

- **Function Evaluation** $\mathsf{Eval}(f, \mathsf{tag}_i, x_i, r_{i,1}, I, \rho_1)$**:** The function evaluation algorithm takes as input the function f to be evaluated, the set I of participating parties, P_i's private input x_i, the randomness $r_{i,1}$ which P_i used to generate its input encoding, and the input encoding transcript ρ_1, and does the following:

1. Parse $\rho_1 = \{\hat{m}_{k,1}, \mathsf{com}_k, \mathsf{nmc}_k, \tau_k, \mathsf{zk}_{1,k,V}, \mathsf{zk}_{1,k,P}\}_{k\in[n]}$ and obtain $(r_{i,\mathsf{SM},1}, r_{i,\mathsf{com}}, r_{i,\mathsf{CCA}}, \sigma_{\mathsf{zk},1,i,V}, \sigma_{\mathsf{zk},1,i,P})$ from $r_{i,1}$.
2. Compute the semi-malicious function evaluation encoding

$$\hat{m}_{i,2} \leftarrow \mathsf{SM.Eval}(f, x_i, r_{i,\mathsf{SM},1}, I, \rho_{\mathsf{sm},1}; PRF_{K_i}(f, I, 1))$$

of the underlying semi-malicious protocol, using the transcript $\rho_{\mathsf{sm},1} = \{\hat{m}_{k,1}\}_{k\in I}$ of the semi-malicious input encodings of all parties from I, where the randomness is chosen using the PRF key committed to during the input encoding phase.
3. Compute a commitment $\mathsf{com}_{i,\hat{m}_{i,2}} \leftarrow \mathsf{NICommit}(\hat{m}_{i,2}; PRF_{K_i}(f, I, 2))$ of the encoding $\hat{m}_{i,2}$ using randomness derived from the PRF key committed to during the input encoding phase.
4. For each P_j, $j \in I \setminus \{i\}$:
   - Compute an opening $\sigma_{i,j,\mathsf{CCA}} \leftarrow \mathsf{Opening}(\tau_j, \mathsf{tag}_i, \mathsf{nmc}_i, (x_i, r_{i,\mathsf{SM},1}, K_i, r_{i,\mathsf{com}}), r_{i,\mathsf{CCA}})$ for the non-malleable-commitment nmc_i with respect to τ_j.
   - Compute a round two ZK prover's message $\mathsf{zk}_{2,i\to j,P} \leftarrow \mathsf{ZKProve}_2(\Phi_{\mathsf{zk},i,j}, W_{\mathsf{zk},i}, \sigma_{\mathsf{zk},1,i,P}, \mathsf{zk}_{1,j,V})$, where $\Phi_{\mathsf{zk},i,j}$ is the circuit SAT instance defined in Figure 2. Here $W_{\mathsf{zk},i} = (x_i, r_{i,\mathsf{SM},1}, K_i, r_{i,\mathsf{com}}, \sigma_{i,j,\mathsf{CCA}}, \hat{m}_{i,2})$ is the witness for generating this prover message.
5. Compute a witness encryption $\mathsf{WE}_i \leftarrow \mathsf{WE.Encrypt}(1^\lambda, \Phi_{\mathsf{WE},i}, r_{\mathsf{com},i,\hat{m}_{i,2}})$ where the circuit $\Phi_{\mathsf{WE},i}$ is described in Figure 3, and the plaintext $r_{\mathsf{com},i,\hat{m}_{i,2}} = PRF_{K_i}(f, I, 2)$ is the opening for $\mathsf{com}_{i,\hat{m}_{i,2}}$.
6. Return $m_{i,2} = (\mathsf{com}_{i,\hat{m}_{i,2}}, \{\mathsf{zk}_{2,i\to j,P}\}_{j\in I\setminus\{i\}}, \mathsf{WE}_i)$.

- **Output Computation** $\mathsf{Output}(\{m_{j,1}, m_{j,2}\}_{j\in I})$**:** The output computation algorithm takes as input the input encoding $m_{j,1}$ and the function evaluation encoding $m_{j,2}$ of every party P_j for $j \in I$ and does the following:

  1. Parse
  $$m_{j,1} = (\hat{m}_{j,1}, \mathsf{com}_j, \mathsf{nmc}_j, \tau_j, \mathsf{zk}_{1,j,v}, \mathsf{zk}_{1,j,p})$$
  and
  $$m_{j,2} = (\mathsf{com}_{j,\hat{m}_{j,2}}, \{\mathsf{zk}_{2,j\to k,P}\}_{k\in I\setminus\{j\}}, \mathsf{WE}_j)$$
  for each $j \in I$.
  2. For each $j, k \in I, j \neq k$:
     - Run $\mathsf{ZKVerify}_2(\Phi_{\mathsf{zk},j,k}, \mathsf{zk}_{1,k,v}, \mathsf{zk}_{1,j,p}, \mathsf{zk}_{2,j\to k,p})$, where $\Phi_{\mathsf{zk},j,k}$ is described in Figure 2. If the verification fails, abort and output $\perp$.
  3. For each $j \in I$:
     - Compute the decryption $r_{\mathsf{com},j,\hat{m}_{j,2}} \leftarrow \mathsf{WE.Decrypt}(\mathsf{WE}_j, W_{\mathsf{WE},j})$ of the opening $r_{\mathsf{com},j,\hat{m}_{j,2}}$ to the commitment $\mathsf{com}_{j,\hat{m}_{j,2}}$, using the witness
     $$W_{\mathsf{WE},j} = (\{\mathsf{zk}_{2,k\to j,P}, \mathsf{com}_{j,\hat{m}_{j,2}}\}_{k\neq j}).$$
     If the decryption fails, abort and output $\perp$.
     - Open $\mathsf{com}_{j,\hat{m}_{j,2}}$ to P_j's semi-malicious function evaluation encoding $\hat{m}_{j,2}$ using $r_{\mathsf{com},j,\hat{m}_{j,2}}$.
  4. Compute the output $y \leftarrow \mathsf{Output}(\{\hat{m}_{j,1}, \hat{m}_{j,2}\}_{j\in I})$ using the values $\hat{m}_{j,2}$ obtained from decrypting the witness encryptions along with the semi-malicious input encodings $\hat{m}_{j,2}$.
  5. Output y.

Correctness. Correctness of the protocol follows directly from correctness of the underlying primitives.

6.1 Proof of Security

In this section we prove that the MrNISC protocol given above satisfies the definition of SPS malicious security from Section 4. Assume that there exists a real-world PPT adversary $\mathcal{A}$ for the MrNISC security game. That is, $\mathcal{A}$ takes as input 1^λ and some auxiliary input z, chooses the number of parties M and the set of honest parties $H \subseteq [M]$, and then interacts with the honest parties in an execution of the protocol by submitting queries of the four types described in Section 4 (i.e., Corrupt Input Encoding, Honest Input Encoding, Honest Computation Encoding, and Corrupt Computation Encoding). We prove security by exhibiting an ideal world adversary $\mathcal{S}$ (referred to as the simulator) which runs in time $T_{\mathcal{S}} = 2^{\lambda^\epsilon}$, and interacts with the ideal functionality $\mathcal{U}$ as described in Section 4, such that the outputs of the corresponding experiments $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{\mathsf{Ideal}}_{\mathcal{S}}(\lambda)$ are indistinguishable.

The Relation $\Phi_{\mathsf{zk},i,j}$

Hardwired: The function f and the set I, P_i's tag tag_i, P_i's CCA non-malleable commitment nmc_i, P_i's perfectly binding commitment com_i, P_i's first round semi-malicious MPC message $\hat{m}_{i,1}$, P_j's string τ_j, P_i's commitment $\mathsf{com}_{i,\hat{m}_{i,2}}$ to its semimalicious evaluation encoding $\hat{m}_{i,2}$, and the transcript $\rho_{\mathsf{sm},1}$ of the semi-malicious input encodings of all parties from I.

Input/Witness: $W_{\mathsf{zk},i} = (x_i, r_{i,\mathsf{SM},1}, K_i, r_{i,\mathsf{com}}, \sigma_{i,j,\mathsf{CCA}}, \hat{m}_{i,2})$.

Computation: Verify the following steps.

1. $\mathsf{Open}(\tau_j, \mathsf{tag}_i, \mathsf{nmc}_i, (x_i, r_{i,\mathsf{SM},1}, K_i, r_{i,\mathsf{com}}), \sigma_{i,j,\mathsf{CCA}}) = 1$
2. $\mathsf{com}_i = \mathsf{NICommit}(1^\lambda, (x_i, r_{i,\mathsf{SM},1}, K_i); r_{i,\mathsf{com}})$
3. $\hat{m}_{i,1} = \mathsf{SM.Encode}(1^\lambda, x_i, r_{i,\mathsf{SM},1})$
4. $\hat{m}_{i,2} = \mathsf{SM.Eval}(f, x_i, r_{i,\mathsf{SM},1}, I, \rho_{\mathsf{sm},1}; PRF_{K_i}(f, I, 1))$
5. $\mathsf{com}_{i,\hat{m}_{i,2}} = \mathsf{NICommit}(1^\lambda, \hat{m}_{i,2}; PRF_{K_i}(f, I, 2))$

Output 1 if all the above checks succeed, otherwise output 0.

Figure 2: The Circuit $\Phi_{\mathsf{zk},i,j}$

The Relation $\Phi_{\mathsf{WE},i}$

Hardwired: The function f, the set I, the set of tags of all parties, P_i's first-round verifier zk messsage $\mathsf{zk}_{1,i,V}$, P_i's string τ_i, the first-round prover zk messages, commitments and semi-malicious encodings $\{\mathsf{zk}_{1,j,P}, \hat{m}_{j,1}, \mathsf{com}_j, \mathsf{nmc}_j\}_{j \in I \setminus \{i\}}$ included in the input encodings of all other parties in I.

Witness:

$$W_{\mathsf{WE},i} = (\{\mathsf{zk}_{2,j\to i,P}, \mathsf{com}_{j,\hat{m}_{j,2}}\}_{j \neq i}).$$

Computation: For every $j \in I \setminus \{i\}$,

1. Let
$$\Phi_{\mathsf{zk},j} = \Phi_{\mathsf{zk},j}[f, I, \mathsf{tag}_j, \mathsf{nmc}_j, \mathsf{com}_j, \hat{m}_{j,1}, \tau_i, \mathsf{com}_{j,\hat{m}_{j,2}}, \rho_{\mathsf{sm},1}]$$
be the circuit described in Figure 2, with the values
$$[f, I, \mathsf{tag}_j, \mathsf{nmc}_j, \mathsf{com}_j, \hat{m}_{j,1}, \tau_i, \mathsf{com}_{j,\hat{m}_{j,2}}, \rho_{\mathsf{sm},1}]$$
hardcoded.
2. Compute $\mathsf{ZKVerify}_2(\Phi_{\mathsf{zk},j}, \mathsf{zk}_{1,i,V}, \mathsf{zk}_{1,j,P}, \mathsf{zk}_{2,j\to i,P})$.

Output 1 if all the above checks succeed, otherwise output 0.

Figure 3: The Relation $\Phi_{\mathsf{WE},i}$

6.1.1 The Simulator

The simulator $\mathcal{S}$ initializes $\mathcal{A}$, and invokes the semi-malicious simulator $\mathsf{SMSim}_1(1^\lambda, \mathcal{C}) \to (\{\hat{m}_{i,1}\}_{i \notin \mathcal{C}}, \sigma_{\mathcal{S}})$ to obtain the honest parties simulated semi-malicious input encodings and the simulator's state. It then responds to $\mathcal{A}$'s queries in the following manner:

- Corrupt Input Encoding: Upon receiving a corrupt input encoding $m_{j,1} = (\hat{m}_{j,1}, \mathsf{com}_j, \mathsf{nmc}_j, \tau_j, \mathsf{zk}_{1,j,v}, \mathsf{zk}_{1,j}$ on behalf of P_j, $j \in \mathcal{C}$, the simulator $\mathcal{S}$ extracts com_j to obtain $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j)$, and submits $(j, \tilde{x}_j)$ to the ideal functionality $\mathcal{U}$ if $\hat{m}_{j,1}$ is honest.
- Honest Input Encoding: Upon receiving this query from $\mathcal{A}$ asking for P_i's input encoding, $\mathcal{S}$ does the following:
  1. Compute a perfectly binding commitment
  $$\mathsf{com}_i = \mathsf{NICommit}(1^\lambda, 0^{|x_i|+|r_{i,\mathsf{SM},1},|+|K_i|}).$$
  2. Compute a CCA-non-malleable commitment
  $$\mathsf{nmc}_i = \mathsf{CCACommit}(1^\lambda, \mathsf{tag}_i, 0^{|x_i|+|r_{i,\mathsf{SM},1},|+|K_i|+|r_{i,\mathsf{com}}|}).$$
  3. Compute a random string $\tau_i \xleftarrow{\$} \{0,1\}^\ell$.
  4. Compute the first round verifier's message and state $(\sigma_{\mathsf{zk},1,i,V}, \mathsf{zk}_{1,i,V}) \leftarrow \mathsf{ZKVerify}_1(1^\lambda)$ and the first round prover message and state $(\sigma_{\mathsf{zk},1,i,P}, \mathsf{zk}_{1,i,P}) \leftarrow \mathsf{ZKProve}_1(1^\lambda)$.
  5. Send $m_{i,1} = (\hat{m}_{i,1}, \mathsf{com}_i, \mathsf{nmc}_i, \tau_i, \mathsf{zk}_{1,i,v}, \mathsf{zk}_{1,i,p})$ to $\mathcal{A}$.
- Honest Computation Encoding: Upon receiving an honest computation encoding query asking for honest party P_i's encoding w.r.t f and I, the simulator does the following.
  1. Compute the extracted value $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}) \leftarrow \mathsf{CCAVal}(\tau_i, \mathsf{tag}_j, \mathsf{nmc}_j)$ of P_j's CCA-non-malleable commitment for each $j \in I \cap \mathcal{C}$.
  2. For each $j \in I \cap \mathcal{C}$, check whether
  $$\hat{m}_{j,1} = \mathsf{SM.Encode}(1^\lambda, \tilde{x}_j; \tilde{r}_{j,\mathsf{SM},1})$$
  and
  $$\mathsf{com}_j = \mathsf{NICommit}(1^\lambda, (\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j); \tilde{r}_{j,\mathsf{com}}),$$
  where $\hat{m}_{j,1}$ is the semi-malicious input encoding sent by P_j, com_j is the perfectly-binding commitment sent by P_j, and $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}$ are the extracted values from before.
  - If both equalities hold for all $j \in I \cap \mathcal{C}$, then the simulator does the following.
    (a) If the simulator has not queried the ideal functionality $\mathcal{U}$ on (f, I) before, then the simulator sends the query (f, I) to $\mathcal{U}$. Upon receiving the function evaluation y, the simulator invokes the semi-malicious simulator $\mathsf{SMSim}_2(\sigma_{\mathcal{S}}, I, f, \{\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \hat{m}_{j,1}\}_{j \in \mathcal{C}}, y) \to \{\hat{m}_{i,2}\}_{i \in I \setminus \mathcal{C}}$ to obtain the simulated honest parties' semi-malicious function evaluation encodings. It then stores $\{\hat{m}_{i,2}\}_{i \in I \setminus \mathcal{C}}$ for use in all future queries. If the simulator has already queried $\mathcal{U}$ on (f, I), then it uses the value $\hat{m}_{i,2}$ which was previously computed.
    (b) Compute a commitment $\mathsf{com}_{i,\hat{m}_{i,2}} \leftarrow \mathsf{NICommit}(\hat{m}_{i,2}; r_{\mathsf{com},i,\hat{m}_{i,2}})$ obtained in the previous step, where $r_{\mathsf{com},i,\hat{m}_{i,2}}$ is freshly chosen randomness.

(c) For each P_j, $j \in I \setminus \{i\}$:
   * Compute a simulated prover's second-round ZK message $\mathsf{zk}_{2,i\to j,P} \leftarrow \mathsf{ZKSim}(\Phi_{\mathsf{zk}}, \sigma_{\mathsf{zk},1,i,P}, \mathsf{zk}_{1,j,}$
(d) Compute a witness encryption $\mathsf{WE}_i \leftarrow \mathsf{WE.Encrypt}(1^\lambda, \Phi_{\mathsf{WE},i}, r_{\mathsf{com},i,\hat{m}_{i,2}})$ where the circuit $\Phi_{\mathsf{WE},i}$ is described in Figure 3, and the plaintext $r_{\mathsf{com},i,\hat{m}_{i,2}}$ is the opening for $\mathsf{com}_{i,\hat{m}_{i,2}}$.
(e) Send $m_{i,2} = (\mathsf{com}_{i,\hat{m}_{i,2}}, \{\mathsf{zk}_{2,i\to j,P}\}_{j\in I\setminus\{i\}}, \mathsf{WE}_i)$ to $\mathcal{A}$.

– If the equalities do not hold for some $j \in I \cap \mathcal{C}$, then the simulator instead does the following:
(a) Compute a commitment $\mathsf{com}_{i,\hat{m}_{i,2}} \leftarrow \mathsf{NICommit}(0^{|\hat{m}_{i,2}|})$.
(b) For each P_j, $j \in I \setminus \{i\}$:
   * Compute a simulated prover's second-round ZK message $\mathsf{zk}_{2,i\to j,P} \leftarrow \mathsf{ZKSim}(\Phi_{\mathsf{zk}}, \sigma_P, \mathsf{zk}_{1,j,V})$.
(c) Compute a witness encryption $\mathsf{WE.CT}_i \leftarrow \mathsf{WE.Encrypt}(1^\lambda, \Phi_{\mathsf{WE},i}, 0^{|r_{i,\mathsf{com}}|})$.
(d) Return $m_{i,2} = (\mathsf{com}_{i,\hat{m}_{i,2}}, \{\mathsf{zk}_{2,i\to j,P}\}_{j\in I\setminus\{i\}}, \mathsf{WE}_i)$.

- Corrupt Computation Encoding: Whenever $\mathcal{A}$ submits a corrupt computation encoding on behalf of corrupted party P_j w.r.t. f and I, the simulator does the following:

1. Compute the extracted value $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}) \leftarrow \mathsf{CCAVal}(\tau_i, \mathsf{tag}_j, \mathsf{nmc}_j)$ of P_j's CCA-non-malleable commitment for each $i \in I \setminus \mathcal{C}$.
2. For each $i \in I \setminus \mathcal{C}$, check if there exists a $j \in I \cap \mathcal{C}$ such that:
   – $\mathsf{ZKVerify}_2(\phi_{\mathsf{zk},j,k}, \mathsf{zk}_{1,i,V}, \mathsf{zk}_{1,j,P}, \mathsf{zk}_{2,j\to i,P})$ verifies, and
   – Steps 2-5 of $\Phi_{\mathsf{zk},j,i}$ do not hold with respect to the extracted values $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j$, $\tilde{r}_{j,\mathsf{com}}$ and the input encoding phase of the protocol. Note that this is checkable in polynomial time given the values $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}$.
3. If there does exist such a j, halt the experiment and output a special abort symbol $\perp^*$.
4. Otherwise, if all parties in I have submitted function evaluation encodings for f, and if all parties' ZK messages have verified correctly the simulator instructs $\mathcal{U}$ to deliver the output y to the honest parties. If any ZK messages verify incorrectly, the simulator instructs $\mathcal{U}$ to deliver the output $\perp$ to the honest parties.

6.1.2 The Hybrids

We prove the indistinguishability between the real and ideal worlds via a sequence of hybrids listed below. We argue indistinguishability between each successive pair of hybrids. The first hybrid $\mathbf{Hybrid}_0$ corresponds to the real world experiment, and the last hybrid $\mathbf{Hybrid}_8$ corresponds to the ideal world experiment.

- **$\mathbf{Hybrid}_0$:** In this hybrid, the simulator performs the real-world experiment $\mathsf{Expt}^{\mathsf{Real}}_{\mathcal{A}}(\lambda)$ with $\mathcal{A}$. That is, the simulator responds to the queries of $\mathcal{A}$ as described in the real world experiment defined in Section 4. At the end of the execution, the output of the hybrid is defined to be $(\mathsf{view}_{\mathcal{A}}, \mathsf{functions\_learned}, \mathsf{honest\_outputs})$.

- **$\mathbf{Hybrid}_1$:** In this hybrid, the simulator behaves identically to the previous hybrid, except for the following difference. Whenever $\mathcal{A}$ submits a Honest Computation Encoding query asking for honest party P_i's encoding w.r.t f and I, the simulator does the following:

1. Compute the extracted value $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}) \leftarrow \mathsf{CCAVal}(\tau_i, \mathsf{tag}_j, \mathsf{nmc}_j)$ of P_j's CCA-non-malleable commitment for each $j \in I \cap \mathcal{C}$.

2. For each $j \in I \cap \mathcal{C}$, check whether

$$\hat{m}_{j,1} = \mathsf{SM.Encode}(1^\lambda, \tilde{x}_j; \tilde{r}_{j,\mathsf{SM},1})$$

and

$$\mathsf{com}_j = \mathsf{NICommit}(1^\lambda, (\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j); \tilde{r}_{j,\mathsf{com}}),$$

where $\hat{m}_{j,1}$ is the semi-malicious input encoding sent by P_j, com_j is the perfectly-binding commitment sent by P_j, and $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}$ are the extracted values from before.

   - If both equalities hold for all $j \in I \cap \mathcal{C}$, then the simulator generates $\mathsf{WE.CT}_i$ in the same way as in $\mathbf{Hybrid}_0$.
   - If the equalities do not hold for some $j \in I \cap \mathcal{C}$, then the simulator instead computes $\mathsf{WE.CT}_i \leftarrow \mathsf{WE.Encrypt}(1^\lambda, \Phi_{\mathsf{WE},i}, 0^{|r_{i,\mathsf{com}}|})$.

Because of the use of CCAVal, this hybrid runs in size $O(T_2)$ and polynomial depth.

- **$\mathbf{Hybrid}_2$:** In this hybrid, the simulator behaves identically to the previous hybrid, except for the following difference. Whenever $\mathcal{A}$ submits a Corrupt Computation Encoding on behalf of corrupted party P_j w.r.t. f and I, the simulator does the following:

  1. Compute the extracted value $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}) \leftarrow \mathsf{CCAVal}(\tau_i, \mathsf{tag}_j, \mathsf{nmc}_j)$ of P_j's CCA-non-malleable commitment for each $i \in I \setminus \mathcal{C}$.
  2. For each $i \in I \setminus \mathcal{C}$, check if there exists a $j \in I \cap \mathcal{C}$ such that:
     - $\mathsf{ZKVerify}_2(\phi_{\mathsf{zk},j,k}, \mathsf{zk}_{1,i,V}, \mathsf{zk}_{1,j,P}, \mathsf{zk}_{2,j\to i,P})$ verifies, and
     - Steps 2-5 of $\Phi_{\mathsf{zk},j,i}$ do not hold with respect to the extracted values $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j$, $\tilde{r}_{j,\mathsf{com}}$ and the input encoding phase of the protocol. Note that this is checkable in polynomial time given the values $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}$.
  3. If there does exist such a j, halt and output a special abort symbol $\perp^*$.

  Because of the use of CCAVal, this hybrid runs in size $O(T_2)$ and polynomial depth.

- **$\mathbf{Hybrid}_3$:** In this hybrid, the simulator behaves identically to the previous hybrid, except for the following difference. Whenever $\mathcal{A}$ submits a Honest Computation Encoding query asking for honest party P_i's encoding w.r.t f and I, the simulator computes P_i's ZK prover's messages $\mathsf{zk}_{2,i\to j,P} \leftarrow \mathsf{ZKSim}(\Phi_{\mathsf{zk}}, \sigma_P, \mathsf{zk}_{1,j,V})$ using the simulator instead of generating the message using the honest prover. This hybrid runs in size $\mathsf{poly}(T_1 + T_2) = \mathsf{poly}(T_2)$ and depth T_1 as we run the ZK Simulator and CCAVal.

- **$\mathbf{Hybrid}_4$:** In this hybrid, the simulator behaves identically to the previous hybrid, except for the following difference. Whenever $\mathcal{A}$ submits a Honest Input Encoding query asking for honest party P_i's first message, the simulator generates $\mathsf{nmc}_i = \mathsf{CCACommit}(1^\lambda, \mathsf{tag}_i, 0^{|x_i|+|r_{i,\mathsf{SM},1},|+|K_i|+|r_{i,\mathsf{com}}|})$. This hybrid runs in the same size and depth as the previous hybrid.

- **$\mathbf{Hybrid}_5$:** In this hybrid, the simulator behaves identically to the previous hybrid, except for the following difference. Whenever $\mathcal{A}$ submits a Honest Input Encoding query asking for honest party P_i's first message, the simulator generates $\mathsf{com}_i = \mathsf{NICommit}(1^\lambda, 0^{|x_i|+|r_{i,\mathsf{SM},1},|+|K_i|})$. This hybrid runs in the same size and depth as the previous hybrid.

- **Hybrid$_6$:** In this hybrid, the simulator behaves identically to the previous hybrid, except for the following difference. Whenever $\mathcal{A}$ submits a Honest Computation Encoding query asking for honest party P_i's encoding w.r.t f and I, the simulator uses true random strings when computing the semi-honest function evaluation and the perfectly binding commitment, instead of using PRF evaluations. In other words, the simulator computes $m_{i,2} \leftarrow \mathsf{SM.Eval}(f, x_i, r_{i,\mathsf{SM},1}, I, \rho_{\mathsf{sm},1}; r)$ and $\mathsf{com}_{i,\hat{m}_{i,2}} \leftarrow \mathsf{NICommit}(\hat{m}_{i,2}; r')$, where r and r' are freshly chosen randomness.

- **Hybrid$_7$:** In this hybrid, the simulator behaves identically to the previous hybrid, except for the following difference. Whenever $\mathcal{A}$ submits a Honest Computation Encoding query asking for honest party P_i's encoding w.r.t f and I, the simulator computes $\mathsf{com}_{i,\hat{m}_{i,2}} \leftarrow \mathsf{NICommit}(0^{|\hat{m}_{i,2}|})$ whenever the equalities checked in the steps for **Hybrid**$_1$ do not hold. This hybrid runs in the same size and depth as the previous hybrid.

- **Hybrid$_8$:** In this hybrid, the simulator behaves identically to the previous hybrid, except for the following differences. During the beginning of the protocol, the simulator invokes the semi-malicious simulator $\mathsf{SMSim}_1(1^\lambda, \mathcal{C}) \rightarrow (\{\hat{m}_{i,1}\}_{i \notin \mathcal{C}}, \sigma_{\mathcal{S}})$ to obtain the honest parties simulated semi-malicious input encodings and the simulator's state.

  - Whenever $\mathcal{A}$ submits an Honest Input Encoding query asking for honest party P_i's input encoding, the simulator uses the simulated encoding $\hat{m}_{i,1}$ generated at the beginning of the protocol as P_i's semi-malicious input encoding.
  - Whenever $\mathcal{A}$ submits a Corrupt Input Encoding query on behalf of P_j, $j \in \mathcal{C}$, the simulator extracts com_j to obtain $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j)$. If P_j's $\hat{m}_{j,1}$ is honestly generated, the simulator submits $(j, \tilde{x}_j)$ to the ideal functionality $\mathcal{U}$.
  - Whenever $\mathcal{A}$ submits an Honest Computation Encoding query asking for honest party P_i's encoding w.r.t f and I, if the equalities checked in **Hybrid**$_1$ hold and the simulator has not queried the ideal functionality $\mathcal{U}$ on (f, I) before, then the simulator sends the query (f, I) to $\mathcal{U}$. Upon receiving the function evaluation y, the simulator invokes the semi-malicious simulator $\mathsf{SMSim}_2(\sigma_{\mathcal{S}}, I, f, \{\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \hat{m}_{j,1}\}_{j \in \mathcal{C}}, y) \rightarrow \{\hat{m}_{i,2}\}_{i \in I \setminus \mathcal{C}}$ to obtain the simulated honest parties' semi-malicious function evaluation encodings. It then stores $\{\hat{m}_{i,2}\}_{i \in I \setminus \mathcal{C}}$ and uses $\hat{m}_{i,2}$ when constructing P_i's response to $\mathcal{A}$'s query. If the simulator has already queried $\mathcal{U}$ on (f, I), then it uses the value $\hat{m}_{i,2}$ which was previously computed. Note that if the equalities checked in **Hybrid**$_1$ do not hold, the simulator does not need to have a $\hat{m}_{i,2}$ message from P_i to respond to $\mathcal{A}$, since it sends a WE of 0.
  - Whenever $\mathcal{A}$ submits a Corrupt Computation Encoding on behalf of corrupted party P_j w.r.t. f and I, if all parties in I have submitted function evaluation encodings for f, and if all parties' ZK messages have verified correctly and the special abort condition has not occurred, the simulator instructs $\mathcal{U}$ to deliver the output y to the honest parties. If any ZK messages verify incorrectly, the simulator instructs $\mathcal{U}$ to deliver the output $\perp$ to the honest parties.

  This hybrid is identical to the behavior of the ideal-world simulator, and runs in size $\mathsf{poly}(T_3)$ and depth T_1.

We now describe indistinguishability between each hybrids. The indistinguishability between **Hybrid**$_0$ and **Hybrid**$_1$ follows from the soundness properties of the SPS ZK protocol and the

security of the Witness Encryption scheme. Because proving this indistinguishability is the most involved, we dedicate a separate section to the proof.

6.1.3 Indistinguishability Between $\mathbf{Hybrid}_0$ and $\mathbf{Hybrid}_1$

Claim 1. *Assuming:*

- $(\mathcal{C}_{\mathsf{WE}}, \epsilon)$*-security for the witness encryption scheme, where* $\mathcal{C}_{\mathsf{WE}}$ *is the class of circuits of size* $p(T_5)$ *for all polynomials* p *and* $\epsilon = 1/T_5$*,*
- *The zero knowledge protocol is* $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$*-statistically sound, where* $\mathcal{C}_{\mathsf{sound}}$ *is the class of circuits of size* $p(T_5)$ *and polynomial depth for all polynomials* p*, and* $\epsilon_{\mathsf{sound},1} = 1/T_4$*, and* $\epsilon_{\mathsf{sound},2}$ *is any negligible function,*
- *The* CCAVal *extraction procedure for the CCA-non-malleable commitment scheme is a circuit of size* T_2 *and polynomial depth, and*
- $T_2 \ll T_4 \ll T_5$*,*

$\mathbf{Hybrid}_0$ *is computationally indistinguishable from* $\mathbf{Hybrid}_1$*.*

We prove this claim via a sequence of subhybrids, which we describe here. Let $q = q(\lambda)$ be a polynomial upper bound on the number of Honest Computation Encoding queries made by $\mathcal{A}$.

- $\mathbf{Hybrid}_{0,0,0}$ is the same as $\mathbf{Hybrid}_0$.
- $\mathbf{Hybrid}_{0,k,r}$ is the same as $\mathbf{Hybrid}_{0,k,r-1}$, except for the following differences. Whenever $\mathcal{A}$ submits its ℓ-th Honest Computation Encoding query asking for honest party P_i's encoding w.r.t f and I, the simulator does the following:
  1. Compute the extracted value $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}) \leftarrow \mathsf{CCAVal}(\tau_i, \mathsf{tag}_j, \mathsf{nmc}_j)$ of P_j's CCA-non-malleable commitment for each $j \in I \cap \mathcal{C}$.
  2. For each $j \in I \cap \mathcal{C}$, check whether
  $$\hat{m}_{j,1} = \mathsf{SM.Encode}(1^\lambda, \tilde{x}_j; \tilde{r}_{j,\mathsf{SM},1})$$
  and
  $$\mathsf{com}_j = \mathsf{NICommit}(1^\lambda, (\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j); \tilde{r}_{j,\mathsf{com}}),$$
  where $\hat{m}_{j,1}$ is the semi-malicious input encoding sent by P_j, com_j is the perfectly-binding commitment sent by P_j, and $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}$ are the extracted values from before.
     - If both equalities hold for all $j \in I \cap \mathcal{C}$, then the simulator generates $\mathsf{WE.CT}_i$ in the same way as in $\mathbf{Hybrid}_0$.
     - If the equalities do not hold for some $j \in I \cap \mathcal{C}$, then if $i \leq k \in [n] \setminus \mathcal{C}$ and if $\ell \leq r$, the simulator instead computes $\mathsf{WE.CT}_i \leftarrow \mathsf{WE.Encrypt}(1^\lambda, \Phi_{\mathsf{WE},i}, 0^{|r_{i,\mathsf{com}}|})$.
- $\mathbf{Hybrid}_{0,n,q}$ is the same as $\mathbf{Hybrid}_1$.

In the following, we denote with $\mathsf{expt}_{\mathcal{A}}^{0,k,r}$ the output of the simulator during $\mathbf{Hybrid}_{0,k,r}$. Note that for all $k \in [n]$, $\mathbf{Hybrid}_{0,k,q} = \mathbf{Hybrid}_{0,k+1,0}$. Thus, to prove Claim 1, it is then sufficient to prove the following claim.

Claim 2. *For all $k \in [n]$ and $r \in [q]$, assuming: Assuming:*

- $(\mathcal{C}_{\mathsf{WE}}, \epsilon)$*-security for the witness encryption scheme, where* $\mathcal{C}_{\mathsf{WE}}$ *is the class of circuits of size* $p(T_5)$ *for all polynomials* p *and* $\epsilon = 1/T_5$*,*
- *The zero knowledge protocol is* $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$*-statistically sound, where* $\mathcal{C}_{\mathsf{sound}}$ *is the class of circuits of size* $p(T_5)$ *and polynomial depth for all polynomials* p*, and* $\epsilon_{\mathsf{sound},1} = 1/T_4$*, and* $\epsilon_{\mathsf{sound},2}$ *is any negligible function,*
- *The* CCAVal *extraction procedure for the CCA-non-malleable commitment scheme is a circuit of size* T_2 *and polynomial depth, and*
- $T_2 \ll T_4 \ll T_5$*,*

$\mathbf{Hybrid}_{0,k,r}$ *is computationally indistinguishable from* $\mathbf{Hybrid}_{0,k,r-1}$*.*

We will rely on several subclaims in order to prove Claim 2. First we introduce some notation.

Assume for the sake of contradiction that there exists an adversary $(\mathcal{A}, \mathcal{D})$ and an index (k, r) such that $\mathcal{A}$ distinguishes between $\mathbf{Hybrid}_{0,k,r-1}$ and $\mathbf{Hybrid}_{0,k,r}$ with non-negligible probability. That is, assume that

$$\left| \Pr\left[\mathcal{D}(\mathsf{expt}_{\mathcal{A}}^{0,k,r}) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{expt}_{\mathcal{A}}^{0,k,r-1}) = 1\right] \right| \geq 1/p(\lambda),$$

for some polynomial p. Fix some $j^* \in \mathcal{C}$, and consider the event that during $\mathbf{Hybrid}_{0,k,r-1}$ or $\mathbf{Hybrid}_{0,k,r}$:

- $\mathcal{A}$ asks for P_k's honest input encoding,
- $\mathcal{A}$ sends corrupted party P_{j^*}'s input encoding to $\mathcal{S}$, where either $\hat{m}_{j,1} \neq \mathsf{SM.Encode}(1^\lambda, \tilde{x}_j; \tilde{r}_{j,\mathsf{SM},1})$ or $\mathsf{com}_j \neq \mathsf{NICommit}(1^\lambda, (\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j); \tilde{r}_{j,\mathsf{com}})$, and
- $\mathcal{A}$'s r-th Honest Computation Encoding query asks for P_k's encoding w.r.t. some (f, I) such that $j^* \in I$.

Define $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}$ and $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}$ to be the same as $\mathsf{expt}_{\mathcal{A}}^{0,k,r}$ and $\mathsf{expt}_{\mathcal{A}}^{0,k,r-1}$, except that whenever the event above does not occur, the simulator outputs a "dummy evaluation", where all parties behave according to the honest input specification, have input 0, and evaluate the constant $f(x_1, \ldots, x_n) = 0$ with $I = [n]$. Fixing the j^* that maximizes the probability of $\mathcal{A}$ distinguishing these two experiments, we then have that

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1\right] \right| \geq 1/p'(\lambda),$$

for some polynomial $p'(\lambda)$.

Define PS_{k,j^*} to be the event that perfect soundness holds in the zero knowledge instance with prover P_{j^*} and verifier P_k which takes place during $\mathbf{Hybrid}_{0,k,\eta}$ for $\eta \in \{r-1, r\}$. Note that since both hybrids are identical up to the r-th Honest Computation Encoding query, this event is well-defined even if η is unspecified.

With this event defined, we can rewrite the probability

$$\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1\right]$$

as the following:

$$\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \Pr\left[\mathsf{PS}_{k,j^*}\right] + \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] \Pr\left[\overline{\mathsf{PS}}_{k,j^*}\right].$$

Claim 3. *Assuming the zero knowledge protocol is* $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$*-sound where* $\mathcal{C}_{\mathsf{sound}}$*,* $\epsilon_{\mathsf{sound},1}$*, and* $\epsilon_{\mathsf{sound},2}$ *are as in Claim 2, it holds that*

$$\Pr\left[\mathsf{PS}_{k,j^*}\right] \geq \epsilon_{\mathsf{sound},1}.$$

Proof. Assume this is not the case. Then we construct a reduction $\mathcal{R}$ to the soundness mode frequency property of the zero knowledge protocol. $\mathcal{R}$ is a circuit of size $\mathsf{poly}(T_2)$ which does the following:

1. Receive $\mathsf{zk}_{1,V}$ from the challenger.
2. Run $\mathsf{expt}_{\mathcal{A}}^{0,k}$, using $\mathsf{zk}_{1,V}$ as part of P_k's input encoding whenever this encoding is requested from $\mathcal{A}$.
3. Whenever $\mathcal{A}$ sends an input encoding on behalf of P_{j^*}, halt and output the $\mathsf{zk}_{1,j^*,P}$ message which is part of P_{j^*}'s input encoding.

By assumption, PS_{k,j^*} holds with probability $< \epsilon_{\mathsf{sound},1}$. This means that $\mathcal{E}(\tau_1, \sigma_{\mathsf{zk},V,k}) = 1$ with probability $< \epsilon_{\mathsf{sound},1}$. Thus $\mathcal{R}$ contradicts $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$-soundness of the zero knowledge protocol. □

Claim 4. *Assuming the zero knowledge protocol is* $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$*-sound where* $\mathcal{C}_{\mathsf{sound}}$*,* $\epsilon_{\mathsf{sound},1}$*, and* $\epsilon_{\mathsf{sound},2}$ *are as in Claim 2, and the extractor* CCAVal *for the CCA-non-malleable commitment scheme is a* T_2*-size circuit, it holds that for all* k *and* r*,*

$$\left|\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right]\right| \leq \epsilon_{\mathsf{sound},2}.$$

Proof. We prove the claim via a $\mathsf{poly}(T_2)$-size reduction to soundness of the zero knowledge protocol. Assume for the sake of contradiction that

$$\left|\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right]\right| > \epsilon_{\mathsf{sound},2}.$$

We construct the reduction $\mathcal{R}$, which behaves as follows:

1. Receive $\mathsf{zk}_{1,V}$ from the challenger.
2. Run $a \leftarrow \widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k}$ using $\mathsf{zk}_{1,k,V} = \mathsf{zk}_{1,V}$ whenever P_k's input encoding is queried, where a is the output of the experiment. Send $\mathsf{zk}_{1,j^*,P}$ to the challenger. Output $\mathcal{D}(a)$.

Note that the probability that $\mathcal{R}$ distinguishes between soundness modes is exactly

$$\left|\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right]\right|,$$

and thus $\mathcal{R}$ contradicts indistinguishability of soundness mode. □

Claim 5. *Assuming the existence of a distinguishing* $\mathcal{A}$ *as before, the zero knowledge protocol is* $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$*-sound where* $\mathcal{C}_{\mathsf{sound}}$*,* $\epsilon_{\mathsf{sound},1}$*, and* $\epsilon_{\mathsf{sound},2}$ *are as in Claim 2, and the extractor* CCAVal *for the CCA-non-malleable commitment scheme is a* T_2*-size circuit, it holds that for all* k *and* r*,*

$$\left|\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \wedge \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \wedge \mathsf{PS}_{k,j^*}\right]\right| \geq \epsilon_{\mathsf{sound},1}/p(\lambda),$$

for some polynomial $p(\lambda)$*.*

Proof. By Claim 3 the left-hand side of the inequality is at least

$$\left| \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right) \cdot \epsilon_{\mathsf{sound},1} \right|.$$

So it suffices to show that

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right| \geq 1/p(\lambda)$$

for some polynomial $p(\lambda)$.

Recall that by assumption we have

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1\right] \right| \geq 1/\mathsf{poly}(\lambda). \tag{1}$$

We can lower-bound the left-hand side of (1) as

$$\begin{aligned} &\left| \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right) \cdot \Pr\left[\mathsf{PS}_{k,j^*}\right] + \right. \\ &\left. \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] \right) \cdot \Pr\left[\overline{\mathsf{PS}}_{k,j^*}\right] \right|, \end{aligned}$$

which by claim Claim 4 is

$$\begin{aligned} &\leq \left| \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right) \cdot \left(\Pr\left[\mathsf{PS}_{k,j^*}\right] + \Pr\left[\overline{PS}_{k,j^*}\right]\right) \right| \\ &+ 2\epsilon_{\mathsf{sound},2} \cdot \Pr\left[\overline{\mathsf{PS}}_{k,j^*}\right]. \end{aligned}$$

(i.e., substitute out $Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right]$ and $\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right]$ for $\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] + \epsilon_{\mathsf{sound},2}$ and $\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] + \epsilon_{\mathsf{sound},2}$, respectively.)

Thus,

$$\left| \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right) \right| \geq 1/\mathsf{poly}(\lambda) - 2\epsilon_{\mathsf{sound},2},$$

which proves the claim. □

Claim 6. *Assuming the "perfect soundness holds during soundness mode" property of the zero knowledge argument, and $(\mathcal{C}_{\mathsf{WE}}, \epsilon)$-security for the witness encryption scheme, where $\mathcal{C}_{\mathsf{WE}}$ is the class of circuits of size $p(T_5)$ for all polynomials p and $\epsilon = 1/T_5$, and $T_5 \gg T_2$, the size of the extraction procedure* CCAVal *for the CCA commitment, it holds that for all k,*

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}) = 1 \wedge \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}) = 1 \wedge \mathsf{PS}_{k,j^*}\right] \right| < \epsilon_{\mathsf{WE}}.$$

Proof. Fix a state state of the experiment just before the r-th Honest Computation Encoding. We show that given such a state where PS_{k,j^*} holds,

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}(\mathsf{state})) = 1\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state})) = 1\right] \right| < \epsilon_{\mathsf{WE}}.$$

We consider two cases. First is the case in which the "dummy evaluation" is triggered. In this case, the output of both $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}(\mathsf{state})$ and $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state})$ are both drawn from exactly the same distribution, and thus

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}(\mathsf{state}_1)) = 1\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state}_1)) = 1\right] \right| = 0.$$

The second case is where the "dummy evaluation" is not triggered, i.e. where the following three conditions are satisfied:

- $\mathcal{A}$ asks for P_k's honest input encoding,
- $\mathcal{A}$ sends corrupted party P_{j^*}'s input encoding to $\mathcal{S}$, where either $\hat{m}_{j,1} \neq \mathsf{SM.Encode}(1^\lambda, \tilde{x}_j; \tilde{r}_{j,\mathsf{SM},1})$ or $\mathsf{com}_j \neq \mathsf{NICommit}(1^\lambda, (\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j); \tilde{r}_{j,\mathsf{com}})$, and
- $\mathcal{A}$'s r-th Honest Computation Encoding query asks for P_k's encoding w.r.t. some (f, I) such that $j^* \in I$.

In this case, the difference between the two experiments is that when responding to the r-th Honest Computation Encoding in $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state})$, the simulator sends $\mathsf{WE.CT}_k \leftarrow \mathsf{WE.Encrypt}(1^\lambda, \Phi_{\mathsf{WE},k}, r_{k,\mathsf{com}})$ to $\mathcal{A}$ on behalf of P_k, whereas in $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}(\mathsf{state})$, the simulator sends $\mathsf{WE.CT}_k \leftarrow \mathsf{WE.Encrypt}(1^\lambda, \Phi_{\mathsf{WE},k}, 0^{|r_{k,\mathsf{com}}|})$. Here $\Phi_{\mathsf{WE},k}$ is the statement in Figure 3.

Assume for the sake of contradiction that

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}(\mathsf{state}_1)) = 1\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state}_1)) = 1\right] \right| \geq \epsilon_{\mathsf{WE}}.$$

WLOG fix the randomness of $\mathcal{A}$ which maximizes this probability. Note that if $\mathcal{A}$ is deterministic this means that state fully determines the statement $\Phi_{\mathsf{WE},k}$.

We build a reduction $\mathcal{R}$ which is of size T_2 and contradicts security of the witness encryption scheme. $\mathcal{R}$ has state hardcoded and does the following:

1. Receive $\mathsf{WE.CT}_k \leftarrow \mathsf{WE.Encrypt}(\Phi_{\mathsf{WE},k}, m)$ from the challenger, where m is either $r_{k,\mathsf{com}}$ or $0^{|r_{k,\mathsf{com}}|}$, and $\Phi_{\mathsf{WE},k}$ is the statement fixed by state and the randomness of $\mathcal{A}$.
2. Run $b \leftarrow \mathcal{D}(\widetilde{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state}_1))$, where $\widetilde{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state}_1)$ is computed in the same way as $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state}_1)$, except using $\mathsf{WE.CT}_k$ as P_k's witness encryption during the r-th Honest Computation Encoding.

If the challenger chooses $m = r_{k,\mathsf{com}}$ then the experiment run by $\mathcal{R}$ is exactly the same as $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r-1}(\mathsf{state})$; if the challenger chooses $m = 0^{|r_{k,\mathsf{com}}|}$ then the experiment is exactly the same as $\widehat{\mathsf{expt}}_{\mathcal{A}}^{0,k,r}(\mathsf{state})$. Note that the statement $\Phi_{\mathsf{WE},k}$ is false because of perfect soundness of the zero knowledge scheme. Thus $\mathcal{R}$ is a size-T_2 machine which distinguishes between two different witness encryptions for the same false statement, thus contradicting security of the witness encryption scheme. □

We now finish the proof of Claim 2 using the three claims proven above.

Proof of Claim 2. We directly achieve a contradiction by applying Claim 5 and Claim 6, along with the fact that $\epsilon_{\mathsf{sound},1} \gg \epsilon_{\mathsf{WE}}$. □

6.1.4 Proving Indistinguishability of the Remaining Hybrids

The proof of indistinguishability between $\mathbf{Hybrid}_1$ and $\mathbf{Hybrid}_2$ is very similar to the previous proof. We include it for the sake of completeness.

Claim 7. *Assuming:*

- *The zero knowledge protocol is $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$-statistically sound, where $\mathcal{C}_{\mathsf{sound}}$ is the class of circuits of size $p(T_5)$ and polynomial depth for all polynomials p, and $\epsilon_{\mathsf{sound},1} = 1/T_4$, and $\epsilon_{\mathsf{sound},2}$ is any negligible function,*

- *The* CCAVal *extraction procedure for the CCA-non-malleable commitment scheme is a circuit of size* T_2 *and polynomial depth, and*
- $T_2 \ll T_4 \ll T_5$,

$\mathbf{Hybrid}_1$ *is computationally indistinguishable from* $\mathbf{Hybrid}_2$.

We prove this claim via a sequence of subhybrids, which we describe here. Let $q = q(\lambda)$ be a polynomial upper bound on the number of Corrupt Computation Encoding queries made by $\mathcal{A}$.

- $\mathbf{Hybrid}_{1,0,0}$ is the same as $\mathbf{Hybrid}_1$.
- $\mathbf{Hybrid}_{1,k,r}$ is the same as $\mathbf{Hybrid}_{1,k,r-1}$, except for the following differences. Whenever $\mathcal{A}$ submits its ℓ-th Corrupt Computation Encoding, on behalf of some corrupted party P_j w.r.t. f and I, then the simulator does the following:
  1. Compute the extracted value $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}) \leftarrow \mathsf{CCAVal}(\tau_i, \mathsf{tag}_j, \mathsf{nmc}_j)$ of P_j's CCA-non-malleable commitment for each $i \in I \setminus \mathcal{C}$.
  2. For each $i \in I \setminus \mathcal{C}$, $i \leq k$, check if there exists a $j \in I \cap \mathcal{C}$ such that:
     - $\mathsf{ZKVerify}_2(\phi_{\mathsf{zk},j,k}, \mathsf{zk}_{1,i,V}, \mathsf{zk}_{1,j,P}, \mathsf{zk}_{2,j\to i,P})$ verifies, and
     - Steps 2-5 of $\Phi_{\mathsf{zk},j,i}$ do not hold with respect to the extracted values $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j$, $\tilde{r}_{j,\mathsf{com}}$ and the input encoding phase of the protocol. Note that this is checkable in polynomial time given the values $\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j, \tilde{r}_{j,\mathsf{com}}$.
  3. If there does exist such a j, then if either $i < k$, or if $i = k$ and $\ell \leq r$, halt and output a special abort symbol $\perp^*$.
- $\mathbf{Hybrid}_{1,n,q}$ is the same as $\mathbf{Hybrid}_2$.

Note that for all $k \in [n]$, $\mathbf{Hybrid}_{1,k,q} = \mathbf{Hybrid}_{1,k+1,0}$. Thus, to prove Claim 7, it is then sufficient to prove the following claim.

Claim 8. *For all* $k \in [n]$ *and* $r \in [q]$*, assuming: Assuming:*

- *The zero knowledge protocol is* $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$*-statistically sound, where* $\mathcal{C}_{\mathsf{sound}}$ *is the class of circuits of size* $p(T_5)$ *and polynomial depth for all polynomials* p*, and* $\epsilon_{\mathsf{sound},1} = 1/T_4$*, and* $\epsilon_{\mathsf{sound},2}$ *is any negligible function,*
- *The* CCAVal *extraction procedure for the CCA-non-malleable commitment scheme is a circuit of size* T_2 *and polynomial depth, and*
- $T_2 \ll T_4 \ll T_5$,

$\mathbf{Hybrid}_{1,k,r}$ *is computationally indistinguishable from* $\mathbf{Hybrid}_{1,k,r-1}$.

We will rely on several subclaims in order to prove Claim 8. First we introduce some notation. In the following, we denote with $\mathsf{expt}_{\mathcal{A}}^{1,k,r}$ the output of the simulator during $\mathbf{Hybrid}_{1,k,r}$.

Assume for the sake of contradiction that there exists an adversary $(\mathcal{A}, \mathcal{D})$ and an index (k, r) such that $\mathcal{A}$ distinguishes between $\mathbf{Hybrid}_{1,k,r-1}$ and $\mathbf{Hybrid}_{1,k,r}$ with non-negligible probability. That is, assume that

$$\left| \Pr\left[\mathcal{D}(\mathsf{expt}_{\mathcal{A}}^{1,k,r}) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{expt}_{\mathcal{A}}^{1,k,r-1}) = 1\right] \right| \geq 1/p(\lambda),$$

for some polynomial p. Fix some $j^* \in \mathcal{C}$, and consider the event that during $\mathbf{Hybrid}_{1,k,r-1}$ or $\mathbf{Hybrid}_{1,k,r}$:

- $\mathcal{A}$ asks for P_k's honest input encoding,
- $\mathcal{A}$ sends corrupted party P_{j^*}'s input encoding to $\mathcal{S}$, and
- $\mathcal{A}$'s r-th Corrupt Computation Encoding query sends P_{j^*}'s computation encoding w.r.t. some (f, I) such that $k \in I$,and the conditions for special abort hold with respect to this encoding.

Define $\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}$ and $\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}$ to be the same as $\mathsf{expt}_{\mathcal{A}}^{1,k,r}$ and $\mathsf{expt}_{\mathcal{A}}^{1,k,r-1}$, except that whenever the event above does not occur, the simulator outputs a "dummy evaluation", where all parties behave according to the honest input specification, have input 0, and evaluate the constant $f(x_1, \ldots, x_n) = 0$ with $I = [n]$. Fixing the j^* that maximizes the probability of $\mathcal{A}$ distinguishing these two experiments, we then have that

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1\right] \right| \geq 1/p'(\lambda),$$

for some polynomial $p'(\lambda)$.

Define PS_{k,j^*} to be the event that perfect soundness holds in the zero knowledge instance with prover P_{j^*} and verifier P_k which takes place during $\mathbf{Hybrid}_{1,k,\eta}$ for $\eta \in \{r-1, r\}$. Note that since both hybrids are identical up to the r-th Corrupt Computation Encoding query, this event is well-defined even if η is unspecified.

With this event defined, we can rewrite the probability

$$\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1\right]$$

as the following:

$$\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \Pr\left[\mathsf{PS}_{k,j^*}\right] + \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] \Pr\left[\overline{\mathsf{PS}}_{k,j^*}\right].$$

Claim 9. *Assuming the zero knowledge protocol is $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$-sound where $\mathcal{C}_{\mathsf{sound}}$, $\epsilon_{\mathsf{sound},1}$, and $\epsilon_{\mathsf{sound},2}$ are as in Claim 8, it holds that*

$$\Pr\left[\mathsf{PS}_{k,j^*}\right] \geq \epsilon_{\mathsf{sound},1}.$$

Proof. Assume this is not the case. Then we construct a reduction $\mathcal{R}$ to the soundness mode frequency property of the zero knowledge protocol. $\mathcal{R}$ is a circuit of size $\mathsf{poly}(T_2)$ which does the following:

1. Receive $\mathsf{zk}_{1,V}$ from the challenger.
2. Run $\mathsf{expt}_{\mathcal{A}}^{1,k}$, using $\mathsf{zk}_{1,V}$ as part of P_k's input encoding whenever this encoding is requested from $\mathcal{A}$.
3. Whenever $\mathcal{A}$ sends an input encoding on behalf of P_{j^*}, halt and output the $\mathsf{zk}_{1,j^*,P}$ message which is part of P_{j^*}'s input encoding.

By assumption, PS_{k,j^*} holds with probability $< \epsilon_{\mathsf{sound},1}$. This means that $\mathcal{E}(\tau_1, \sigma_{\mathsf{zk},V,k}) = 1$ with probability $< \epsilon_{\mathsf{sound},1}$. Thus $\mathcal{R}$ contradicts $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$-soundness of the zero knowledge protocol. □

Claim 10. *Assuming the zero knowledge protocol is* $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$*-sound where* $\mathcal{C}_{\mathsf{sound}}$*,* $\epsilon_{\mathsf{sound},1}$*, and* $\epsilon_{\mathsf{sound},2}$ *are as in Claim 8, and the extractor* CCAVal *for the CCA-non-malleable commitment scheme is a* T_2*-size circuit, it holds that for all* k *and* r*,*

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] \right| \leq \epsilon_{\mathsf{sound},2}.$$

Proof. We prove the claim via a $\mathsf{poly}(T_2)$-size reduction to soundness of the zero knowledge protocol. Assume for the sake of contradiction that

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] \right| > \epsilon_{\mathsf{sound},2}.$$

We construct the reduction $\mathcal{R}$, which behaves as follows:

1. Receive $\mathsf{zk}_{1,V}$ from the challenger.
2. Run $a \leftarrow \widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}$ using $\mathsf{zk}_{1,k,V} = \mathsf{zk}_{1,V}$ whenever P_k's input encoding is queried, where a is the output of the experiment. Send $\mathsf{zk}_{1,j^*,P}$ to the challenger. Output $\mathcal{D}(a)$.

Note that the probability that $\mathcal{R}$ distinguishes between soundness modes is exactly

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] \right|,$$

and thus $\mathcal{R}$ contradicts indistinguishability of soundness mode. □

Claim 11. *Assuming the existence of a distinguishing* $\mathcal{A}$ *as before, the zero knowledge protocol is* $(\mathcal{C}_{\mathsf{sound}}, \epsilon_{\mathsf{sound},1}, \epsilon_{\mathsf{sound},2})$*-sound where* $\mathcal{C}_{\mathsf{sound}}$*,* $\epsilon_{\mathsf{sound},1}$*, and* $\epsilon_{\mathsf{sound},2}$ *are as in Claim 8, and the extractor* CCAVal *for the CCA-non-malleable commitment scheme is a* T_2*-size circuit, it holds that for all* k *and* r*,*

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \wedge \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \wedge \mathsf{PS}_{k,j^*}\right] \right| \geq \epsilon_{\mathsf{sound},1}/p(\lambda),$$

for some polynomial $p(\lambda)$*.*

Proof. By Claim 9 the left-hand side of the inequality is at least

$$\left| \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right) \cdot \epsilon_{\mathsf{sound},1} \right|.$$

So it suffices to show that

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right| \geq 1/p(\lambda)$$

for some polynomial $p(\lambda)$.

Recall that by assumption we have

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1\right] \right| \geq 1/\mathsf{poly}(\lambda). \tag{2}$$

We can lower-bound the left-hand side of (2) as

$$\begin{aligned} &\left| \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right) \cdot \Pr\left[\mathsf{PS}_{k,j^*}\right] + \right. \\ &\left. \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right] \right) \cdot \Pr\left[\overline{\mathsf{PS}}_{k,j^*}\right] \right|, \end{aligned}$$

which by claim Claim 10 is

$$\leq \left| \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right) \cdot \left(\Pr\left[\mathsf{PS}_{k,j^*}\right] + \Pr\left[\overline{PS}_{k,j^*}\right]\right) \right| + 2\epsilon_{\mathsf{sound},2} \cdot \Pr\left[\overline{\mathsf{PS}}_{k,j^*}\right].$$

(i.e., substitute out $Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right]$ and $\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \mid \overline{\mathsf{PS}}_{k,j^*}\right]$ for $\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] + \epsilon_{\mathsf{sound},2}$ and $\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] + \epsilon_{\mathsf{sound},2}$, respectively.)

Thus,

$$\left| \left(\Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \mid \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \mid \mathsf{PS}_{k,j^*}\right] \right) \right| \geq 1/\mathsf{poly}(\lambda) - 2\epsilon_{\mathsf{sound},2},$$

which proves the claim. □

Claim 12. *Assuming the "perfect soundness holds during soundness mode" property of the zero knowledge argument, , it holds that for all k,*

$$\left| \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r}) = 1 \wedge \mathsf{PS}_{k,j^*}\right] - \Pr\left[\mathcal{D}(\widehat{\mathsf{expt}}_{\mathcal{A}}^{1,k,r-1}) = 1 \wedge \mathsf{PS}_{k,j^*}\right] \right| = 0.$$

Proof. This follows directly from the perfect soundness mode of the ZK argument scheme. □

We now finish the proof of Claim 8 using the three claims proven above.

Proof of Claim 8. We directly achieve a contradiction by applying Claim 11 and Claim 12. □

Claim 13. *Assuming the ZK argument scheme satisfies $(\mathcal{C}_{\mathcal{S}}, \mathcal{C}_{\mathsf{zk}}, \epsilon_{\mathcal{S}})$-adaptive reusable statistical zero knowledge, where $\mathcal{C}_{\mathcal{S}}$ is the class of circuits of size $\mathsf{poly}(T_1)$ and depth T_1 (i.e. the simulator runs in size $\mathsf{poly}(T_1)$ and depth T_1), and $\mathcal{C}_{\mathsf{zk}}$ is the class of circuits of size $p(T_3)$ for all polynomials p, and $\epsilon_{\mathcal{S}}$ is any negligible function, and the CCA extractor CCAVal is a circuit of size T_2, where $T_2 \ll T_3$, then for any polynomial time MPC adversary $\mathcal{A}$ and unbounded distinguisher $\mathcal{D}$, we have*

$$|\Pr[\mathcal{D}(\mathbf{Hybrid}_2) = 1] - \Pr[\mathcal{D}(\mathbf{Hybrid}_3) = 1]| < \mathsf{negl}(\lambda)$$

for some negligible negl.

Proof. This can be done by introducing $|[n] \setminus \mathcal{C}| \cdot n$ intermediate hybrids. We index them as $\mathbf{Hybrid}_{2,i,j,r}$ for $i \in [n] \setminus \mathcal{C}$, $j \in [n]$, and $r \in [q]$, where $q = q(\lambda)$ is a polynomial upper bound on the number of Honest Computation Encoding queries made by $\mathcal{A}$. $\mathbf{Hybrid}_{2,i_2,j_2,r_2}$ comes after $\mathbf{Hybrid}_{2,i_1,j_1,r_1}$ if $i_1 > i_2$, and otherwise if $i_1 = i_2$, $j_1 > j_2$, and otherwise if $i_1 = i_2$, $j_1 = j_2$, $r_1 > r_2$. $\mathbf{Hybrid}_{2,i,j,r}$ is exactly the same as the same as the previous hybrid except that if $i \in [n] \setminus \mathcal{C}$, $\mathsf{zk}_{2,i \to j,P}$ in the r-th Honest Computation Encoding query for P_i is now generated by running $\mathsf{ZKSim}(\Phi_{\mathsf{zk},i}, \mathsf{zk}_{1,i,P}, \mathsf{zk}_{1,j,V})$. Note that the final hybrid in the series is exactly the same as $\mathbf{Hybrid}_3$. To prove the claim, it suffices to show indistinguishability between each successive pair of subhybrids.

Assume for the sake of contradiction that $(\mathcal{A}, \mathcal{D})$ distinguishes between two successive subhybrids $\mathbf{Hybrid}_{2,i_2,j_2,r_2}$ and $\mathbf{Hybrid}_{2,i_1,j_1,r_1}$. We then construct a reduction $(\mathcal{R}, \mathcal{D})$ which breaks the statistical ZK property of the zero knowledge protocol. $\mathcal{R}$ is a circuit of size $\mathsf{poly}(T_2)$ and depth T_1 and does the following:

1. Receive $\mathsf{zk}_{1,P}$ from the challenger.

2. Run $\mathbf{Hybrid}_{2,i_1,j_1}$ with $\mathcal{A}$, using $\mathsf{zk}_{1,i_2,P} = \mathsf{zk}_{1,P}$ (i.e. use the challenger's round-one zk prover's message as the round-one prover's message for P_{i_2} as part of its input encoding.

3. When $\mathcal{A}$ asks for the r-th honest computation encoding from P_i, send the message $\mathsf{zk}_{1,j_2,V}$ along with the statement $\Phi_{\mathsf{zk},i_2,j_2}$ and the witness W_{zk,i_2} to the challenger.

4. Receive $\mathsf{zk}_{2,P}$ from the challenger.

5. Generate P_{i_2}'s honest computation encoding in the same way as in $\mathbf{Hybrid}_{2,i_1,j_1,r_1}$ except using $\mathsf{zk}_{2,i_2\to j_2,P} = \mathsf{zk}_{2,P}$ (i.e. use the challenger's round-two zk prover's message as the round-two prover's message for P_{i_2} in the protocol between P_{i_2} and P_{j_2}.

6. Output the result of the experiment.

If the challenger sends an honest proof of $\Phi_{\mathsf{zk},i_2,j_2}$ to $\mathcal{R}$, then the output of $\mathcal{R}$ is identical to $\mathbf{Hybrid}_{2,i_1,j_1,r_1}$; otherwise, if the challenger simulates the proof, then the output of $\mathcal{R}$ is identical to $\mathbf{Hybrid}_{2,i_2,j_2,r_2}$. Note that $\mathcal{R}$ has size $\ll T_3$; thus by assumption $(\mathcal{R},\mathcal{D})$ contradicts $(\mathcal{C}_{\mathcal{S}},\mathcal{C}_{\mathsf{zk}},\epsilon_{\mathcal{S}})$-statistical zero knowledge of the zero knowledge protocol. □

Claim 14. *Assuming that the CCA non-malleable commitment scheme satisfies $(\mathcal{C},\epsilon)$-CCA security (Definition 17), where $\mathcal{C}$ contains all circuits of size $\mathsf{poly}(T_1)$ where T_1 is the size of the ZK simulator, and ϵ is any negligible function, we have that for any polynomial time MPC adversary $(\mathcal{A},\mathcal{D})$:*

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_3]=1]-\Pr[\mathcal{D}[\mathbf{Hybrid}_4]=1]|\leq \mathsf{negl}(\lambda),$$

for some negligible negl.

Proof. We show this by constructing intermediate hybrids $\mathbf{Hybrid}_{3,i}$ for $i\in[n]$. We define $\mathbf{Hybrid}_{3,i}$ to be identical to the previous hybrid except that if P_i is honest, nmc_i is generated as a non-malleable commitment of all zero string with tag tag_i during the Honest Input Encoding query. Note that $\mathbf{Hybrid}_{3,0}$ is identical to $\mathbf{Hybrid}_3$ and $\mathbf{Hybrid}_{3,n}$ is identical to $\mathbf{Hybrid}_4$. We show that for any two intermediate hybrids $\mathbf{Hybrid}_{3,i-1}$ and $\mathbf{Hybrid}_{3,i}$, it holds that for any polynomial time distingusher $\mathcal{D}$:

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_{3,i-1}]=1]-\Pr[\mathcal{D}[\mathbf{Hybrid}_{3,i}]=1]|\leq \mathsf{negl}(\lambda)$$

The only difference is how nmc_i is generated. If the advantage in distinguishing between the two is more than $\frac{1}{\mathsf{poly}(\lambda)}$ for some polynomial poly, then, we can create a reduction $\mathcal{R}$ that runs in time $\mathsf{poly}(T_1)$ and breaks the security of the receiver-assisted CCA commitment scheme with the same advantage. Here is how the reduction works:

- $\mathcal{R}$ submits $\mathsf{tag}^* = \mathsf{tag}_i$ to the CCA challenger.
- It runs the adversary $(\mathcal{A},\mathcal{D})$ as in $\mathbf{Hybrid}_{3,i-1}$.
- $\mathcal{R}$ generates $\mathsf{nmc}_{i'}$ for all $i'\in[n]\setminus\mathcal{C}$ and $i'\neq i$ as in $\mathbf{Hybrid}_{3,i-1}$.
- For all $P_{i'}$, $i'\in[n]\setminus\mathcal{C}$, $\mathcal{R}$ sends a τ-query to the CCA challenger, and uses the response as the string $\tau_{i'}$ given the input encoding for $P_{i'}$.
- When $\mathcal{R}$ receives the Honest Input Encoding query for P_i, it sends $\alpha_0=(x_i,r_{i,\mathsf{SM}},K_i,r_{i,\mathsf{com}})$ and $\alpha_1=0^{|x_i,r_{i,\mathsf{SM}},K_i,r_{i,\mathsf{com}}|}$ to the challenger of the non-malleable commitment. It gets a response nmc^* which is a commitment with respect to the tag tag_i. It it either a commitment of α_0 or α_1. The reduction uses this as nmc_i when constructing P_i's input encoding.

- Whenever $\mathbf{Hybrid}_{3,i-1}$ needs to extract a CCA commitment nmc_j w.r.t. tag_j and some honest $\tau_{i'}$, $\mathcal{R}$ sends a query $(\tau_{i'}, \mathsf{tag}_j, \mathsf{nmc}_j)$, and uses the response as the extracted value.
- Finally it outputs whatever $\mathcal{D}$ outputs.

Note that if nmc^* is a commitment of α_0, then the view is identical as in $\mathbf{Hybrid}_{3,i}$, otherwise it is as in $\mathbf{Hybrid}_{3,i-1}$. The reduction runs in time polynomial in T_1, since excluding the simulation for ZK rest of the steps are polynomial time. Further, the CCAVal algorithm is never invoked for the challenge tag tag_i. Thus if $\mathcal{D}$ distinguishes between the two cases with probability $\frac{1}{\mathsf{poly}(\lambda)}$, then, it must win in the CCA non-malleable commitment security game with the same advantage.

This proves the claim. □

Claim 15. *Assume that the perfectly binding commitment scheme is hiding against adversaries of size* $\mathsf{poly}(T_2)$*, where* T_2 *is the size of the* CCAVal *circuit. Then we have that*

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_4] = 1] - \Pr[\mathcal{D}[\mathbf{Hybrid}_5] = 1]| \leq \mathsf{negl}(\lambda)$$

for some negligible negl*.*

Proof. We show this by constructing intermediate hybrids $\mathbf{Hybrid}_{4,i}$ for $i \in [n]$. We define $\mathbf{Hybrid}_{4,i}$ to be identical to the previous hybrid except that if P_i is honest, com_i is generated as a non-malleable commitment of all zero string during the Honest Input Encoding query. Note that $\mathbf{Hybrid}_{4,0}$ is identical to $\mathbf{Hybrid}_4$ and $\mathbf{Hybrid}_{4,n}$ is identical to $\mathbf{Hybrid}_5$. We show that for any two intermediate hybrids $\mathbf{Hybrid}_{4,i-1}$ and $\mathbf{Hybrid}_{4,i}$, it holds that for any polynomial time distingusher $\mathcal{D}$:

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_{4,i-1}] = 1] - \Pr[\mathcal{D}[\mathbf{Hybrid}_{4,i}] = 1]| \leq \mathsf{negl}(\lambda)$$

The only difference is how com_i is generated. Assume there is an $(\mathcal{A}, \mathcal{D})$ where the distinguishing advantage between the two is more than $\frac{1}{\mathsf{poly}(\lambda)}$ for some polynomial poly. Then we can create a reduction $\mathcal{R}$ that runs in time $\mathsf{poly}(T_2)$, and contradicts hiding of the commitment scheme. The reduction works as follows:

- It runs the adversary $(\mathcal{A}, \mathcal{D})$ as in $\mathbf{Hybrid}_{4,i-1}$.
- The reduction generates com_j for all $j \in [n] \setminus \mathcal{C}$ and $j \neq i$ as in $\mathbf{Hybrid}_{4,i-1}$.
- It sends $\alpha_0 = (x_i, r_{i,\mathsf{SM}}, K_i)$ and $\alpha_1 = 0^{|x_i, r_{i,\mathsf{SM}}, K_i|}$ to the challenger of the perfectly binding commitment. It gets a response com^*. It it either a commitment of α_0 or α_1. The reduction uses this in constructing P_i's input encoding
- The reduction runs the rest of the experiment exactly the same as $\mathbf{Hybrid}_{4,i-1}$.

Note that if com^* is a commitment of α_0, then the view is identical as in $\mathbf{Hybrid}_{4,i}$, otherwise it is as in $\mathbf{Hybrid}_{4,i-1}$. The reduction runs in time polynomial in T_2. Thus if $\mathcal{D}$ distinguishes between the two cases with probability $\frac{1}{\mathsf{poly}(\lambda)}$, then, it contradicts hiding of the perfectly binding commitment scheme against adversaries of size $\mathsf{poly}(T_2)$.

This proves the claim. □

Claim 16. *Assume that the PRF is secure against adversaries of size* $\mathsf{poly}(T_2)$*, where* T_2 *is the size of the* CCAVal *circuit. Then we have that*

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_5] = 1] - \Pr[\mathcal{D}[\mathbf{Hybrid}_6] = 1]| \leq \mathsf{negl}(\lambda)$$

for some negligible negl*.*

Proof. We show this by constructing intermediate hybrids $\mathbf{Hybrid}_{5,i}$ for $i \in [n]$. We define $\mathbf{Hybrid}_{5,i}$ to be identical to the previous hybrid except that if P_i is honest, then during any the HONEST COMPUTATION ENCODING query for P_i the hybrid generates $\hat{m}_{i,2}$ and $\mathsf{com}_{i,\hat{m}_{i,2}}$ using true randomness instead of the PRF evaluations. Note that $\mathbf{Hybrid}_{5,0}$ is identical to $\mathbf{Hybrid}_5$ and $\mathbf{Hybrid}_{5,n}$ is identical to $\mathbf{Hybrid}_6$. We show that for any two intermediate hybrids $\mathbf{Hybrid}_{5,i-1}$ and $\mathbf{Hybrid}_{5,i}$, it holds that for any polynomial time distinguisher $\mathcal{D}$:

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_{5,i-1}] = 1] - \Pr[\mathcal{D}[\mathbf{Hybrid}_{5,i}] = 1]| \leq \mathsf{negl}(\lambda)$$

Assume there is an $(\mathcal{A}, \mathcal{D})$ where the distinguishing advantage between the two is more than $\frac{1}{\mathsf{poly}(\lambda)}$ for some polynomial poly. Then we can create a reduction $\mathcal{R}$ that runs in time $\mathsf{poly}(T_2)$, and contradicts security of the PRF. The reduction works as follows:

- It runs the adversary $(\mathcal{A}, \mathcal{D})$ as in $\mathbf{Hybrid}_{5,i-1}$.
- The reduction generates computation encodings for all $j \in [n] \setminus \mathcal{C}$ and $j \neq i$ as in $\mathbf{Hybrid}_{5,i-1}$.
- Whenever a HONEST COMPUTATION ENCODING query is made requesting P_i's encoding, $\mathcal{R}$ makes two queries to the PRF oracle at indices $(f, I, 1)$ and $(f, I, 2)$, receiving strings r_1 and r_2. It then uses r_1 as the randomness when computing $\hat{m}_{i,2}$, and uses r_2 as the randomness when computing $\mathsf{com}_{i,\hat{m}_{i,2}}$.
- The reduction runs the rest of the experiment exactly the same as $\mathbf{Hybrid}_{5,i-1}$.

Note that if the oracle is supplying PRF values, then the view is identical as in $\mathbf{Hybrid}_{5,i}$. If the oracle is supplying true random values, the view is as in $\mathbf{Hybrid}_{5,i-1}$. The reduction runs in time polynomial in T_2. Thus if $\mathcal{D}$ distinguishes between the two cases with probability $\frac{1}{\mathsf{poly}(\lambda)}$, then, it contradicts security of the PRF against adversaries of size $\mathsf{poly}(T_2)$.

This proves the claim. □

Claim 17. *Assume that the perfectly binding commitment scheme is secure against adversaries of size* $\mathsf{poly}(T_2)$*, where* T_2 *is the size of the* CCAVal *circuit. Then we have that*

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_6] = 1] - \Pr[\mathcal{D}[\mathbf{Hybrid}_7] = 1]| \leq \mathsf{negl}(\lambda)$$

for some negligible negl*.*

Proof. We show this by constructing intermediate hybrids $\mathbf{Hybrid}_{6,i,r}$ for $i \in [n]$, $r \in [q]$, where q is a (polynomial) upper bound on the total number of HONEST COMPUTATION ENCODING queries that $\mathcal{A}$ makes. We define $\mathbf{Hybrid}_{6,i,r}$ to be identical to the previous hybrid except that if P_i is honest, then during the r-th HONEST COMPUTATION ENCODING query for P_i, the hybrid computes $\mathsf{com}_{i,\hat{m}_{i,2}} \leftarrow \mathsf{NICommit}(0^{|\hat{m}_{i,2}|})$ whenever the equalities checked in the steps for $\mathbf{Hybrid}_1$ do not hold. Note that $\mathbf{Hybrid}_{6,i,q} = \mathbf{Hybrid}_{6,i+1,0}$, $\mathbf{Hybrid}_{6,1,0} = \mathbf{Hybrid}_6$, and $\mathbf{Hybrid}_{6,n,q} = \mathbf{Hybrid}_7$. We show that for any two intermediate hybrids $\mathbf{Hybrid}_{6,i,r-1}$ and $\mathbf{Hybrid}_{6,i,r}$, it holds that for any polynomial time distinguisher $\mathcal{D}$:

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_{6,i,r-1}] = 1] - \Pr[\mathcal{D}[\mathbf{Hybrid}_{6,i,r}] = 1]| \leq \mathsf{negl}(\lambda)$$

Assume there is an $(\mathcal{A}, \mathcal{D})$ where the distinguishing advantage between the two is more than $\frac{1}{\mathsf{poly}(\lambda)}$ for some polynomial poly. Then we can create a reduction $\mathcal{R}$ that runs in time $\mathsf{poly}(T_2)$, and contradicts security of the PRF. First, fix the randomness used in all rounds before the r-th

Honest Computation Encoding made to P_i. Let (f, I) be this r-th query. In particular, this fixes whether or not the equalities check in the steps for $\mathbf{Hybrid}_1$ hold w.r.t. P_i, f and I. It also fixes P_i's semi-honest MrNISC message $\hat{m}_{i,2}$ which it computes when computing its r-th honest computation encoding. If we fix the randomness such that the distinguishing advantage is maximized, then the distinguishing advantage must still be polynomial in λ. This means that the equalities must not hold, otherwise the two hybrids are identical.

The reduction $\mathcal{R}$ then works as follows. It plays a game with a commitment challenger, which either gives a commitment to $\hat{m}_{i,2}$ or $0^{|\hat{m}_{i,2}|}$ $\mathcal{R}$ does the following:

- It runs the adversary $(\mathcal{A}, \mathcal{D})$ as in $\mathbf{Hybrid}_{6,i,r-1}$, with the randomness fixed as described above.
- When $\mathcal{A}$ submits the r-th *Honest Computation Encoding*, $\mathcal{R}$ queries the challenger to get com, which it then uses as $\mathsf{com}_{i,\hat{m}_{i,2}}$ when generating its response on behalf of P_i.
- $\mathcal{R}$ runs the rest of the experiment in the same way as $\mathbf{Hybrid}_{6,i,r-1}$.

Note that if the challenger sends $\mathcal{R}$ a commitment to $\hat{m}_{i,2}$, then the view is identical to that in $\mathbf{Hybrid}_{6,i,r-1}$. If the challenger sends a commitment to $0^{|\hat{m}_{i,2}|}$, the view is as in $\mathbf{Hybrid}_{6,i,r}$. The reduction runs in time polynomial in T_2. Thus if $\mathcal{D}$ distinguishes between the two cases with probability $\frac{1}{\mathsf{poly}(\lambda)}$, then, it contradicts the hiding of the perfectly binding commitment scheme against adversaries of size $\mathsf{poly}(T_2)$.

This proves the claim. □

Claim 18. *Assuming semi-malicious security of the underlying semi-malicious protocol holds against* $\mathsf{poly}(T_3)$*-time adversaries, where* T_3 *is the size of the* $\mathsf{NICommit}$ *commitment scheme extractor,*

$$|\Pr[\mathcal{D}[\mathbf{Hybrid}_5] = 1] - \Pr[\mathcal{D}[\mathbf{Hybrid}_6] = 1]| \leq \mathsf{negl}(\lambda).$$

Proof. Assume for the sake of contradiction that there exists an adversary $(\mathcal{A}, \mathcal{D})$ which distinguishes between the two hybrids with non-negligible probability. We build a reduction $(\mathcal{R}, \mathcal{D})$ to the semi-malicious security of the underlying semi-malicious protocol. $\mathcal{R}$ runs in time T_3, and behaves as follows:

1. Whenever $\mathcal{A}$ submits an Honest Input Encoding query asking for honest party P_i's input encoding, $\mathcal{R}$ sends the same Honest Input Encoding query for P_i to the semimalicious challenger. It then uses the responds $\hat{m}_{i,1}$ when computing P_i's input encoding for $\mathcal{A}$.
2. Whenever $\mathcal{A}$ submits a Corrupt Input Encoding query on behalf of P_j, $j \in \mathcal{C}$, $\mathcal{R}$ extracts com_j to obtain $(\tilde{x}_j, \tilde{r}_{j,\mathsf{SM},1}, \tilde{K}_j)$. If P_j's $\hat{m}_{j,1}$ is honestly generated, the $\mathcal{R}$ submits $\hat{m}_{j,1}$ to the challenger as P_j's message, along with the explanation $(j, \tilde{x}_j, r_{j,\mathsf{SM},1})$.
3. Whenever $\mathcal{A}$ submits an Honest Computation Encoding query asking for honest party P_i's encoding w.r.t f and I, if the equalities checked in $\mathbf{Hybrid}_1$ hold, $\mathcal{R}$ sends the same Honest Computation Encoding to the challenger. It uses the (semi-malicious) response $\hat{m}_{i,2}$ when constructing P_i's (malicious) response to $\mathcal{A}$'s query. If the checks do not hold, $\mathcal{R}$ responds to $\mathcal{A}$ without querying the challenger.
4. Whenever $\mathcal{A}$ submits a Corrupt Computation Encoding on behalf of corrupted party P_j w.r.t. f and I, if all parties in I have submitted function evaluation encodings for f, and if all parties' ZK messages have verified correctly and the special abort condition has not occurred, $\mathcal{R}$ sends the underlying semi-honest message $\hat{m}_{j,2}$ to the challenger. If any ZK messages verify incorrectly, $\mathcal{R}$ sends the "abort encoding" to the challenger instead.

5. At the end of the experiment, output the view of $\mathcal{A}$.

If the challenger enacts the real-world semi-malicious game, then the output of $\mathcal{R}$ along with the honest parties' outputs are identical to the output of $\mathbf{Hybrid}_7$. If the challenger enacts the ideal-world game, then the outputs are identical to the output of $\mathbf{Hybrid}_8$. Thus by assumption, $(\mathcal{R}, \mathcal{D})$ distinguishes between the real and the ideal world, contradicting semi-malicious security of the underlying semi-malicious MrNISC against time-T_3 adversaries.

□

7 Our Receiver-Assisted One-Round CCA Commitments

7.1 Overview

Our construction relies on a recent work of Khurana [Khu21] which we explain next. The main technical contribution of [Khu21] is a tag-amplification procedure. Starting from a one-round CCA commitment scheme (without any receiver-assisted randomness) for small tags (say tags lie in $[T']$ where $T' = \log\log\lambda$), they build a receiver-assisted scheme with a much larger tag space (say supporting tags in $[T]$ where $T = T'^{\Omega(T')}$). This transformation can be applied once again on top of the resulting receiver-assisted one-round CCA scheme to get a receiver-assisted scheme supporting a super-polynomial number of tags. Thus, a constant number of applications suffice to construct a scheme for $2^{\Omega(\lambda)}$ tags. At the base level, schemes supporting $\log\log\lambda$ tags are known[3] from well-studied assumptions, such as time-lock puzzles and one-way permutations [LPS20, BL18] or quantum hardness of LWE and classical hardness of DDH [KK19].

In more detail, the tag-amplification procedure makes use of a base commitment scheme $\mathsf{nmc} = (\mathsf{CCACommit}, \mathsf{CCAVal})$ for small tags in $[T']$ where $T' = \log\log\lambda$, an indistinguishability obfuscator iO, a public-key encryption scheme PKE with dense public keys, a non-interactive witness indistinguishable proofs NIWI, a puncturable PRF PPRF, and a one-way permutation $\mathsf{OWP} : \{0,1\}^{\ell_{\mathsf{OWP}}} \to \{0,1\}^{\ell_{\mathsf{OWP}}}$ (actually a one-way function with verifiable range suffices, but we describe using a permutation for simplicity).

In the scheme, the tag space of the resulting scheme consists of subsets of $[T']$ of size exactly $T'/2$. Thus, $T = \binom{T'}{T'/2}$. The idea is the following: to commit to a message m with respect to $\mathsf{tag} \in [T]$, parse tag as $(t_1, \ldots, t_{T'/2})$ where each $t_i \in [T']$. Then, the commitment simply consists of an iO obfuscation of the program described in Figure 4, where pk and the PPRF key K_{PPRF} are freshly sampled by the committer and hardwired into the program.

The scheme builds on the intuition (borrowed from [Khu21]) that such a tag amplification can be done non-interactively without receiver's assistance if one-message zero-knowledge existed (in the plain model, without a crs). The idea is that one can generate a non-malleable commitment of message m with respect to tag by computing $c_1, \ldots, c_{T'/2}$ where each $c_i = \mathsf{nmc.CCACommit}(t_i, m)$, as also done in the description of the circuit in Figure 4, and simply computing a ZK proof π to the fact that all c_i's were generated by committing a consistent message m. The resulting commitment consists of $(c_1, \ldots, c_{T'/2}, \pi)$. This argument can be made formal if such a ZK existed. Roughly, since we use a ZK, the soundness property ensures that the CCAVal oracle is queried on well-formed commitments, all committed to same value m, and therefore in the reduction $\mathsf{nmc.CCAVal}(\star)$ is only needed to run for some tags, and not all. At the same time the zero knowledge property along with the hiding property of nmc can be used to argue indistinguishability of the challenge commitments.

[3] Although, these base schemes satisfy security with one-tag restriction, [Khu21] give a method, which is used first to boost it to security for a same number of tags, but without this restriction. We will follow the same approach.

The Circuit $G[t_1, \ldots, t_{T'/2}, m, K_{\mathsf{PPRF}}, \mathsf{pk}]$

Hardwired: Tags $(t_1, \ldots, t_{T'/2}) \in [T']^{T'/2}$, Message m and PPRF key K_{PPRF}, public key pk,

Input: $\rho \in \{0,1\}^{\ell_{\mathsf{OWP}}}$

Computation:

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \rho)$ and parse $r = (r_1, r_{2,1}, \ldots, r_{2,T'/2}, r_3)$. Compute:
   - $c_0 = \mathsf{PKE.Enc}(\mathsf{pk}, 0^{\ell_{\mathsf{OWP}}}; r_1)$,
   - For $i \in [T'/2]$, compute $c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t_i, m; r_{2,i})$,
   - Compute $\pi = \mathsf{NIWI.P}(X, W; r_3)$ where $X = (c_0, c_1, \ldots, c_{T'/2})$ and $W = (m, \{r_{2,i}\}_{i \in [T']})$ are so that $(X, W) \in L_G$ for the language L_G defined below.
2. Output $(c_0, c_1, \ldots, c_{T'/2}, \pi)$.

Language $L_G = L_{G_1} \vee L_{G,2}$:

$$L_{G,1} = \{(c_0, c_1, \ldots, c_{T'/2}) \mid \exists x \in \{0,1\}^{\ell_{\mathsf{OWP}}} \ s.t. \ c_0 = \mathsf{Enc}(\mathsf{pk}, x) \wedge \mathsf{OWP}(x) = \rho\}$$
$$L_{G,2} = \{(c_0, c_1, \ldots, c_{T'/2}) \mid \exists m \ s.t. \ \forall i \in [T'/2], \ c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t_i, m)\}$$

Figure 4: The Circuit $G[t_1, \ldots, t_{T'/2}, m, k_{\mathsf{PPRF}}]$

Since one-message zero-knowledge does not exist, the hope is that generating commitments this way can be useful to revive this approach. A receiver can evaluate the program on a randomly chosen input ρ to compute $(c_0, c_1, \ldots, c_{T'/2}, \pi)$. The point is that if π verifies, then this means that unless c_0 is an encryption of $\mathsf{OWP}^{-1}(\rho)$, $c_1, \ldots, c_{T'/2}$ must be well formed nmc commitments of the same message m. This ensures that the CCAVal oracle is invoked on only well formed commitments, which was important in the previous proof. Secondly, to argue security one can go "input-by-input". For $\alpha \in [0, 2^{\ell_{\mathsf{OWP}}} - 1]$, we switch the obfuscated circuit to commit to m_1 as opposed to m_0 when the input $\rho \leq \alpha$. To do so, we need to hardwire non-uniformly $\beta = \mathsf{OWP}^{-1}(\alpha)$ into the reduction at each hybrid, so that the reduction can generate c_0 by encrypting β and use it to prove NIWI. For the security arguments to go through it requires that the public-key encryption, NIWI and the base commitments are more secure with advantage at least $2^{-\ell_{\mathsf{OWP}}}$.

This yields the following contradiction. On one hand, public-key encryption needs to be more secure than the OWP to argue security. On the other hand, we need OWP to be secure against the time it takes to break c_0 to extract a pre-image of ρ chosen by the challenger to show that the adversary does not query the CCAVal algorithm on non-well formed commitments.

Nevertheless, [Khu21] observed that if the receiver randomness ρ's are chosen *after* declaring the set of commitments that would be queried to the CCAVal oracle, this issue can be handled via the following clever idea: the reduction can guess the secret-keys $\{\mathsf{sk}_i\}$ associated with the public keys $\{\mathsf{pk}_i\}$ used in the commitment programs $\{P_i\}$ chosen by the adversary. If a program P_i produces "bad" outputs $(c_0, c_1, \ldots, c_{T'/2}, \pi)$ on a large fraction of points ρ, then one can recover inverses of $\mathsf{OWP}^{-1}(\rho)$ using a non-uniformly fixed secret key sk. This gives a non-uniform reduction to the security of OWP.

There is another reason why the construction of [Khu21] only provides security if the receiver randomness is chosen only after all commitments P_i for which $\mathsf{CCAVal}(\star)$ may be queried are

displayed the adversary. The reason is that on one hand nmc needs to be more secure than OWP to argue indistinguishability, on the other hand, OWP needs to be secure against the circuit that can run nmc.CCAVal($\star$) to handle CCAVal queries in the reduction which can be done in any order. This problem does not arise if adversary outputs commitments P_i for which CCAVal is queried, up front: the reduction never really has to actually run CCAVal to recover inverses to OWP challenge as the commitments P_i are already revealed.

In summary, the above idea only works in the setting where receiver randomness is chosen after the adversary displays all commitments for which CCAVal may be queried, but unfortunately it fails in our setting, where receiver randomness can be sampled uniformly, at any point, any number of times the adversary demands. Our main idea is to introduce a new axis of hardness. We use quantum-classical tradeoffs. We replace the public key encryption with a perfectly binding quantum polynomial-time breakable commitment scheme and rely on an nmc schemes where nmc.CCAVal runs in quantum polynomial time.

How does this help? Consider that the OWP is quantum secure, then if the adversary submits a commitment P and a query $\mathsf{CCAVal}(\rho, \mathsf{tag}, P)$ such that $P[\rho] = (c_0, c_1, \ldots, c_{T'/2}, \pi)$ where π verifies, and $c_1, \ldots, c_{T'/2}$ are not consistent and well formed, then one can form an efficient quantum adversary that runs in time polynomial in the time of the adversary that breaks the OWP security. The idea is that for this to happen c_0 must be a commitment of $\mathsf{OWP}^{-1}(\rho)$ due to perfect soundness of NIWI. And, then, c_0 can be simply inverted by running a quantum polynomial time extractor of the commitment. Note that the reduction also needs to respond to CCAVal queries while interacting with the adversary, but those can also be run in quantum polynomial time.

In the classical world, commitments to compute c_0 and nmc, NIWI and other primitives can be made to be more secure than OWP to go input by input and argue security of the commitment scheme. This brings us to one last issue. Except we are not aware of a quantum secure one-way permutation, from well-studied assumptions. To deal with this issue, we observe that we could have also used a quantum secure collision resistant hash function, where the keys are randomly chosen (such as known via the small-integer soultion/LWE problems). In this case c_0 will be used to commit to a collision for the hash function.

Technical Remarks. Before we describe our construction, we mention a technical issue. The underlying non-malleable commitments such as [BL18, KK19] have to issues that we have to deal with:

- They satisfy security with one-tag restriction, and,
- To support $\Omega(\log \log \lambda)$, assuming subexponential security of the underlying assumptions, these schemes are only quasi-polynomially secure. Thus, our transfomation should work with those parameters.

Our transformation below actually works with a quasi-polynomially secure base commitment. For the first problem, we follow as in [Khu21], and take an nmc with one-tag restriction and convert it to a receiver assisted scheme for a same number of tags, but without this restriction. This transformation is extremely similar to our tag-amplification and we sketch this in Section 7.3. We now describe our construction.

7.2 Our Tag-Amplification Transformation

Our construction is essentially identical to the construction provided by [Khu21] except for a few important changes, which helps us to address the shortcomings in the scheme of [Khu21], mentioned above.

Used Primitives. We make use of the following primitives and instantiated with the following parameters. These instantiated parameters for the primitives we use are loose for what we require.

Receiver-assisted one-round CCA commitments: As a starting point, we make use of a receiver-assisted one-round CCA commitment aCCA' with security parameter $\lambda_{\mathsf{aCCA}'}$ for small tag space $T'(\lambda_{\mathsf{aCCA}'})$. The tag-space $T'(\lambda_{\mathsf{aCCA}'})$ is at least $\underbrace{\log \ldots \log(\lambda_{\mathsf{aCCA}'})}_{O(1) \text{ times}}$ and at most $\lambda^{O(1)}$. We now describe the circuit class against which security holds and other properties involved:

- The scheme satisfies $(\mathcal{C}_{\mathsf{aCCA}'}, \epsilon_{\mathsf{aCCA}'})$-security where $\mathcal{C}_{\mathsf{aCCA}'}$ consists of all circuits of size polynomial in $\frac{1}{\epsilon_{\mathsf{aCCA}'}}$ where $\epsilon_{acc'} = 2^{\lambda^{c_1 (\log\log \lambda_{\mathsf{aCCA}'})^{-1}}}$ where $c_1 > 0$ is some constant.
- $\mathsf{aCCA}'.\mathsf{CCAVal}(\star)$ runs in quantum polynomial time.
- Let $\ell_{\mathsf{aCCA}'}(\lambda_{\mathsf{aCCA}'})$ be the length of the string τ chosen by the receiver,

Resulting Primitive. At the end of a single step of this transformation, we will build the scheme aCCA scheme with security parameter λ. We set $\lambda_{\mathsf{aCCA}'} = \lambda$. After a single step transformation, the resulting scheme will be secure against adversaries of size polynomial in $s_{\mathsf{aCCA}} = 2^{\lambda^{c_2 (\log\log \lambda)^{-1}}}$ with advantage bounded by $\epsilon_{\mathsf{aCCA}} = 2^{-\lambda^{c_2 (\log\log \lambda)^{-1}}}$ for some other constant $c_2 > 0$. $\mathsf{aCCA}.\mathsf{CCAVal}(\star)$ will also be quantum polynomial time implementable. The resulting tag space will be $T(\lambda)$ where $T = T'^{\Omega(T'/2)}$. As a result of applying the procedure constant number of times, we get a super-polynomial number of tags. At the base level, this can be instantiated by taking the scheme of [LPS20, BL18], which satisfies CCA security with one-tag restriction, and then applying the transformation as given in Section 7.3 to give a receiver assisted scheme aCCA' without this restriction. The scheme in [LPS20, BL18] can be instantiated using iterated squaring assumption and the DDH (or SXDH over bilinear maps) assumptions both of which are polynomial time quantum broken.

Perfectly-Binding Quantum Extractable Commitment: We require a perfectly biding commitment scheme NICom, which is extractable in quantum polynomial time. Further, it takes as input λ_{NICom}, and guarantees $2^{-\lambda_{\mathsf{NICom}}}$ indistinguishability against adversaries of size polynomial in $2^{\lambda_{\mathsf{NICom}}}$. Such commitments can be built using subexponential hardness of DDH

Quantum-Secure Collision Resistant Hash Functions with random keys: We require a sub-exponentially secure family of hash functions $\{\mathcal{H}_{\lambda_h} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}\}_{\lambda_h \in \mathbb{N}}$, where the key space is $\mathcal{K} = \{0,1\}^{\ell_{h_{key}}}$, input space is $\mathcal{X} = \{0,1\}^{\ell_{h_{inp}}}$, and the output space is $\mathcal{Y} = \{0,1\}^{\ell_{h_{out}}}$. Above $\ell_{h_{inp}}, \ell_{h_{out}}, \ell_{h_{key}}$ are polynomials in λ_h. We require the following additional properties:

- For every key $K \in \mathcal{K}$ there exists unequal $x, x' \in \mathcal{X}$ such that $\mathcal{H}(K, x) = \mathcal{H}(K, x')$.
- When $K \leftarrow \mathcal{K}$, then for any quantum algorithm running in time polynomial in 2^{λ_h}, the advantage in finding the collisions is bounded by $2^{-\lambda_h}$

We set λ_h as follows. Let λ_h be such that $\ell_{h_{key}} = \lambda^{c_3 (\log\log \lambda)^{-1}}$ where $c_3 = c_1/100$. This means that $\lambda_h = \lambda^{c_3' (\log\log \lambda)^{-1}}$ for some $c_3' < c_3$. In the resulting scheme, c_2 can be arbitrary constant less than c_3'.

Indistinguishability Obfuscation: We require an indistinguishability Obfuscator iO. This

scheme uses λ_{iO} as the security parameter and is secure against adversaries of size $2^{\lambda_{\mathsf{iO}}}$ with advantage $2^{-\lambda_{\mathsf{iO}}}$. Such a primitive can be built using well-studied assumptions as shown in [JLS21a, JLS21b].

PUNCTURABLE PRF: We require a puncturable PRF, $\mathsf{PPRF} = (\mathsf{Puncture}, \mathsf{Eval})$. Assume the length of the key is randomly chosen of length $\ell_{\mathsf{PPRF}}(\lambda_{\mathsf{PPRF}})$ where λ_{PPRF} is its security parameter. The length of the output is some polynomial implicit in the scheme. We assume that the PPRF is secure against adversaries of size polynomial in $2^{\lambda_{\mathsf{PPRF}}}$ with a maximum advantage of $2^{-\lambda_{\mathsf{PPRF}}}$.

NIWI: We require a non-interactive witness indistinguishable proof NIWI for NP, that is secure against adversaries of size polynomial in $2^{\lambda_{\mathsf{NIWI}}}$ with advantage bounded by $2^{-\lambda_{\mathsf{NIWI}}}$. This primitive can be built assuming subexponentially hard SXDH over Bilinear Maps.

We set $\lambda_{\mathsf{iO}} = \lambda_{\mathsf{NIWI}} = \lambda_{\mathsf{PPRF}} = \lambda_{\mathsf{NICom}}$ as a large enough polynomial. In particular, setting $2^{\lambda_{\mathsf{iO}}} \gg \mathsf{Time}(\mathsf{aCCA}'.\mathsf{CCAVal}(\star)) \cdot 2^{\lambda}$ suffices.

Our final observation is that all the primitives described above exist from the following primitives listed under our theorem statement.

Theorem 3. *Assume that the following assumptions hold:*

- *A subexponentially secure indistinguishability obfuscator exists,*
- *A quantum polynomial time breakable time lock puzzle as in Definition 6 exist,*
- *LWE is subexponentially secure against quantum adversaries of subexponential size,*
- *SXDH is subexponentially secure against adversaries of subexponential size,*

then, there exist a receiver assisted one round CCA commitment scheme (as in Definition 14) supporting a super-polynomial number of tags. The scheme is secure against adversaries of size polynomial in $2^{\lambda^{c(\log\log\lambda)^{-1}}}$ for some $c > 0$.

Construction. We now give our construction. We first define the tag space T. As in [Khu21], we will set $T = \binom{T'}{T'/2}$ which is precisely equal to the number of unique subsets of $[T']$ of size $[T'/2]$. Let ϕ be a polynomial time computable bijective map that takes as input $\mathsf{tag} \in [T]$, and outputs a unique subset $\{t_1, \ldots, t_{T'/2}\}$ of $[T']$ of size $T'/2$. These subsets are unique upto permutation. We assume that they are sorted in ascending order.

$\mathsf{aCCA.CCACommit}(\mathsf{tag}, m; r)$: Compute the following steps.

- Compute $\phi(\mathsf{tag}) = (t_1, \ldots, t_{T'/2})$. Sample a PPRF key $K_{\mathsf{PPRF}} \leftarrow \{0,1\}^{\ell_{\mathsf{PPRF}}}$,
- Compute $\widetilde{G} \leftarrow \mathsf{iO}(G[t_1, \ldots, t_{T'/2}, m, K_{\mathsf{PPRF}}])$ by obfuscating the circuit described in Figure 5. Output $\widetilde{G}$.

$\mathsf{aCCA.Opening}(\tau, \mathsf{tag}, \widetilde{G}, m, r)$: Compute the following steps.

- Parse $\tau = (\rho, \rho')$ where $\rho \in \{0,1\}^{\ell_{h_{key}}}$ and $\rho' \in \{0,1\}^{\ell_{\mathsf{aCCA}'}}$,
- Compute $\phi(\mathsf{tag}) = (t_1, \ldots, t_{T'/2})$,
- Check if $\widetilde{G} = \mathsf{aCCA.CCACommit}(\mathsf{tag}, m; r)$. Abort if its not the case. Derive the PPRF key K_{PPRF} used in code of G described in Figure 5.
- Compute $\widetilde{G}[\rho] = (c_0, c_1, \ldots, c_{T'/2}, \pi)$,

- From the code of Figure 5, use the PPRF key k_{PPRF} to derive r'_i as in the code such that $c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t_i, m; r'_i)$. Compute and output $\sigma_i = \mathsf{aCCA}'.\mathsf{Opening}(\rho', t_i, c_i, m, r'_i)$ for $i \in [T'/2]$.

$\mathsf{aCCA.Open}(\tau, \mathsf{tag}, \widetilde{G}, m, \sigma)$: Compute the following steps.

- Parse $\tau = (\rho, \rho')$ where $\rho \in \{0,1\}^{\ell_{h_{key}}}$ and $\rho' \in \{0,1\}^{\ell_{\mathsf{aCCA}'}}$,
- Compute $\phi(\mathsf{tag}) = (t_1, \ldots, t_{T'/2})$ and $\sigma = (\sigma_1, \ldots, \sigma_{T'/2})$,
- Compute $\widetilde{G}[\rho] = (c_0, c_1, \ldots, c_{T'/2}, \pi)$ and verify π using $\mathsf{NIWI.Vf}$ for the language described in Figure 5. Abort if the proof does not verify,
- Output 1 if for every $i \in [T'/2]$, $\mathsf{aCCA}'.\mathsf{Open}(\rho', t_i, c_i, m, \sigma_i)$. Output $\perp$ otherwise.

The Circuit $G[t_1, \ldots, t_{T'/2}, m, K_{\mathsf{PPRF}}]$

Hardwired: Tags $(t_1, \ldots, t_{T'/2}) \in [T']^{T'/2}$, Message m and PPRF key K_{PPRF},

Input: $\rho \in \{0,1\}^{\ell_{h_{key}}}$

Computation:

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \rho)$ and parse $r = (r_1, r_{2,1}, \ldots, r_{2,T'/2}, r_3)$. Compute:
   - $c_0 = \mathsf{NICom}(0^{\ell_{h_{inp}}}; r_1)$,
   - For $i \in [T'/2]$, compute $c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t_i, m; r_{2,i})$,
   - Compute $\pi = \mathsf{NIWI.P}(X, W; r_3)$ where $X = (c_0, c_1, \ldots, c_{T'/2})$ and $W = (m, \{r_{2,i}\}_{i \in [T']})$ are so that $(X, W) \in L_G$ for the language L_G defined below.
2. Output $(c_0, c_1, \ldots, c_{T'/2}, \pi)$.

Language $L_G = L_{G_1} \vee L_{G,2}$:

$$L_{G,1} = \{(c_0, c_1, \ldots, c_{T'/2}) \mid \exists x \neq x' \in \{0,1\}^{\ell_{h_{in}}} \ s.t. \ \mathcal{H}(\rho, x) = \mathcal{H}(\rho, x') \wedge c_0 = \mathsf{NICom}(x, x')\}$$
$$L_{G,2} = \{(c_0, c_1, \ldots, c_{T'/2}) \mid \exists m \ s.t. \ \forall i \in [T'/2], \ c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t_i, m)\}$$

Figure 5: The Circuit $G[t_1, \ldots, t_{T'/2}, m, k_{\mathsf{PPRF}}]$

We now argue various properties involved. The correctness of opening is immediate due to the correctness of opening of the underlying commitment scheme aCCA' and correctness and completeness of other primitives involved. To argue the extraction property, we now describe the $\mathsf{aCCA.CCAVal}$ algorithm.

$\mathsf{aCCA.CCAVal}(\tau, \mathsf{tag}, \widetilde{G})$: Compute the following steps.

- Parse $\tau = (\rho, \rho')$ where $\rho \in \{0,1\}^{\ell_{h_{key}}}$ and $\rho' \in \{0,1\}^{\ell_{\mathsf{aCCA}'}}$,
- Compute $\phi(\mathsf{tag}) = (t_1, \ldots, t_{T'/2})$,
- Compute $\widetilde{G}[\rho] = (c_0, c_1, \ldots, c_{T'/2}, \pi)$ and verify π using $\mathsf{NIWI.Vf}$ for the language described in 5. Abort if the proof does not verify,

- Otherwise output m if $m = \mathsf{aCCA}'.\mathsf{CCAVal}(\rho', t_1, c_1) = \ldots = \mathsf{aCCA}'.\mathsf{CCAVal}(\rho', t_{T'/2}, c_{T'/2})$.

The extraction property then follows immediately from the extraction property of the underlying aCCA' scheme. The idea is that in the last step, if $m = \mathsf{aCCA}'.\mathsf{CCAVal}(\rho', t_1, c_1) = \ldots = \mathsf{aCCA}'.\mathsf{CCAVal}(\rho', t_{T'/2}, c_{T'/2})$, then, there exists openings $\sigma_1, \ldots, \sigma_{T'/2}$, that opens $(c_1, \ldots, c_{T'/2})$ to m due to the extraction property of aCCA'. Similarly, the reverse is also true.

We now move onto the security proof.

7.2.1 Security Proof

The security proof can be structured by giving indistinguishable hybrids. The first one corresponds to the game where the challenger computes $\mathsf{aCCA.CCACommit}(\mathsf{tag}^*, m_b)$ for a random b, where as the last hybrid is independent of b. We describe the first hybrid elaborately, where as in the later hybrids, we just describe the change.

Hybrid_0 : In this hybrid,

1. The challenger manages a list L that is initially empty. The contents of the list are visible to the adversary at all stages.
2. The adversary sends a challenge tag $\mathsf{tag}^* \in \mathcal{T}_\lambda$.
3. The adversary submits queries of the following kind in an adaptive manner:
   (a) Adversary can ask for arbitrary polynomially many τ-query. Challenger samples $\tau' \leftarrow \{0,1\}^{\ell_{\mathsf{aCCA}}}$ and appends τ' to L.
   (b) Adversary can ask for an abitrary polynomially many $(\tau, \mathsf{tag}, \mathsf{P})$-query for any $\tau \in L$, any $\mathsf{tag} \neq \mathsf{tag}^*$, and any commitment P. The challenger computes $\mathsf{CCAVal}(\tau, \mathsf{tag}, \mathsf{P})$ and sends the result to the adversary.
4. The adversary submits two messages m_0, m_1 of equal length. The challenger samples $b \leftarrow \{0,1\}$, and computes $\mathsf{P}^* \leftarrow \mathsf{aCCA.CCACommit}(\mathsf{tag}^*, m_b)$. The adversary gets P^* from the challenger.
5. The adversary repeats Step 3.
6. Finally, the adversary outputs a guess $b' \in \{0,1\}$. The experiment outputs 1 if $b' = b$ and 0 otherwise.

$\mathsf{Hybrid}_{1,j\in[0,Q]}$: This hybrid is the same as the previous hybrid, except that we modify how the CCAVal queries corresponding to j^{th} τ query is responded. Recall that the challenger maintains a list L, and every time adversary makes a τ query, a randomly sampled τ is added to this list. In this hybrid, let τ_j be the sampled τ the j^{th} $\tau-$query made by the adversary. In this hybrid we replace how CCAVal query is responded for $\mathsf{CCAVal}(\tau_j, \mathsf{tag}, \mathsf{P})$ for $\mathsf{tag} \neq \mathsf{tag}*$ and $\tau_j \in L$. The new code is defined as follows.

$\mathsf{aCCA.CCAVal}^*(\tau_j, \mathsf{tag}, \mathsf{P})$: Compute the following steps.

- Parse $\tau_j = (\rho, \rho')$ where $\rho \in \{0,1\}^{\ell_{h_{key}}}$ and $\rho' \in \{0,1\}^{\ell_{\mathsf{aCCA}'}}$,
- Compute $\phi(\mathsf{tag}) = (t_1, \ldots, t_{T'/2})$ and $\phi(\mathsf{tag}^*) = (t^*_1, \ldots, t^*_{T'/2})$,
- Since $\mathsf{tag} \neq \mathsf{tag}'$, there must exist a first index $i \in [T'/2]$ such that $t_i \neq \{t^*_1, \ldots, t^*_{T'/2}\}$.
- Compute $\mathsf{P}[\rho] = (c_0, c_1, \ldots, c_{T'/2}, \pi)$ and verify π using $\mathsf{NIWI.Vf}$ for the language described in Figure 5. Abort if the proof does not verify,

- Otherwise output m where $m = \mathsf{aCCA}'.\mathsf{CCAVal}(\rho', t_i, c_i)$.

Note that $\mathsf{Hybrid}_{1,0}$ is the same as Hybrid_0. We now show that $\mathsf{Hybrid}_{1,j}$ is indistinguishable to $\mathsf{Hybrid}_{1,j+1}$. We show that if there is an adversary with size polynomial in 2^{λ_h} that distinguishes these hybrids with probability p, then there exists a (quantum) reduction that is runs in time polynomial in $\mathsf{poly}(2^{\lambda_h})$, and wins in the collision resistant hash function security game with probability p. Thus, showing that $p < 2^{-\lambda_h}$. We show our reduction.

- Reduction proceeds by maintaining a list L honestly,
- It generates all τ queries honestly, except that it for the $(j+1)^{th}$ query, it sets $\tau_{j+1} = (\rho, \rho')$ where ρ is received from the challenger of the hash function and ρ' is sampled randomly by the challenger.
- It answers CCAVal queries for every τ_i for $i \leq j$ using CCAVal^* (this is well defined because adversary does not use tag^*). It can be answered in quantum polynomial time as $\mathsf{aCCA}'.\mathsf{CCAVal}$ can be implemented in quantum polynomial time.
- It answers CCAVal queries for every τ_i for $i > j+1$ using CCAVal. These queries can be answered in quantum polynomial time.
- For τ_{j+1} it does the following. Assume that the query is for $\mathsf{CCAVal}(\tau_{j+1}, \mathsf{tag}, \mathsf{P})$ for $\mathsf{tag} \neq \mathsf{tag}^*$. Then, do the following:
  - Run $\mathsf{P}[\rho] = (c_0, c_1, \ldots, c_{T'/2}, \pi)$.
  - Output $\perp$ if π does not verify. If it does, break open c_0 in quantum polynomial time to recover x, x'. Check if $\mathcal{H}(\rho, x) = \mathcal{H}(\rho, x')$ for $x \neq x'$. If this is true, output x, x' as the answer to the hash function challenger.
  - Otherwise, output $\mathsf{aCCA}'.\mathsf{CCAVal}(\rho', t_1, c_1)$ to the adversary.

Observe that because $\mathsf{aCCA}'.\mathsf{CCAVal}$ runs in quantum polynomial time, the reduction runs in quantum time polynomial in s_{aCCA}. Further observe, if $\mathcal{A}$ observes a difference between $\mathsf{Hybrid}_{1,j}$ and $\mathsf{Hybrid}_{1,j+1}$, then there must be a query of the form $\mathsf{CCAVal}(\tau_{j+1}, \mathsf{tag}, \mathsf{P})$ that produces different outputs. Parsing $\tau_{j+1} = (\rho, \rho')$, and $\mathsf{P}[\rho] = (c_0, c_1, \ldots, c_{T'/2}, \pi)$, this means that π verifies, but in the least there exists two indices i_1, i_2 such that $\mathsf{aCCA}'.\mathsf{CCAVal}(\rho', t_{i_1}, c_{i_1}) \neq \mathsf{aCCA}'.\mathsf{CCAVal}(\rho', t_{i_2}, c_{i_2})$. By soundness of NIWI, it means that c_0 must be a commitment of a collision for the hash key ρ. Thus, the reduction will succeed at that point.

Thus, we have the following claim.

Lemma 1. *Assuming that $\mathcal{H}$ is a secure against $\mathsf{poly}(2^{\lambda_h})$ sized quantum circuits, NIWI is sound, and aCCA' satisfies perfect correctness/extraction properties, we have that for any adversary of size $\mathsf{poly}(s_{\mathsf{aCCA}})$:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}_{1,j}) = 1] - \Pr[\mathcal{A}(\mathsf{Hybrid}_{1,j+1}) = 1]| \leq 2^{-\lambda_h}$$

for $j \in [0, Q-1]$.

We now describe a series of hybrids. For $\alpha \in [0, 2^{\ell_{h_{key}}}]$.

$\mathsf{Hybrid}_{2,\alpha}$: This hybrid is the same as the previous hybrid, except that in order to generate

The Circuit $G_1[t_1, \ldots, t_{T'/2}, m_b, m_0, \alpha, K_{\mathsf{PPRF}}]$

Hardwired: Tags $(t_1, \ldots, t_{T'/2}) \in [T']^{T'/2}$, Messages m_b and m_0, PPRF key K_{PPRF}, and $\alpha \in [0, 2^{\ell_{h_{key}}}]$.

Input: $\rho \in \{0,1\}^{\ell_{h_{key}}}$

Computation: The computation can be divided into two cases.
Case: $\rho < \alpha$

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \rho)$ and parse $r = (r_1, r_{2,1}, \ldots, r_{2,T'/2}, r_3)$.
2. Compute $c_0 = \mathsf{NICom}(0^{\ell_{h_{inp}}}; r_1)$ and for $i \in [T'/2]$, compute $c_i = \mathsf{aCCA'.CCACommit}(t_i, m_0; r_{2,i})$,
3. Compute $\pi = \mathsf{NIWI.P}(X, W; r_3)$ where $X = (c_0, c_1, \ldots, c_{T'/2})$ and $W = (m_0, \{r_{2,i}\}_{i \in [T']})$ are so that $(X, W) \in L_G$ for the language L_G defined below.
4. Output $(c_0, c_1, \ldots, c_{T'/2}, \pi)$.

Case: $\rho \geq \alpha$

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \rho)$ and parse $r = (r_1, r_{2,1}, \ldots, r_{2,T'/2}, r_3)$.
2. Compute $c_0 = \mathsf{NICom}(0^{\ell_{h_{inp}}}; r_1)$ and for $i \in [T'/2]$, compute $c_i = \mathsf{aCCA'.CCACommit}(t_i, m_b; r_{2,i})$,
3. Compute $\pi = \mathsf{NIWI.P}(X, W; r_3)$ where $X = (c_0, c_1, \ldots, c_{T'/2})$ and $W = (m_b, \{r_{2,i}\}_{i \in [T']})$ are so that $(X, W) \in L_G$ for the language L_G defined below.
4. Output $(c_0, c_1, \ldots, c_{T'/2}, \pi)$.

Language $L_G = L_{G_1} \vee L_{G,2}$:

$$L_{G,1} = \big\{(c_0, c_1, \ldots, c_{T'/2}) \mid \exists x \neq x' \in \{0,1\}^{\ell_{h_{in}}} \ s.t.\ \mathcal{H}(\rho, x) = \mathcal{H}(\rho, x') \wedge c_0 = \mathsf{NICom}(x, x')\big\}$$
$$L_{G,2} = \big\{(c_0, c_1, \ldots, c_{T'/2}) \mid \exists m \ s.t.\ \forall i \in [T'/2],\ c_i = \mathsf{aCCA'.CCACommit}(t_i, m)\big\}$$

Figure 6: The Circuit $G_1[t_1, \ldots, t_{T'/2}, m_b, m_0, \alpha, k_{\mathsf{PPRF}}]$

P^*, we obfuscate the circuit in Figure 6. Namely, compute $\phi(\mathsf{tag}^*) = (t_1^*, \ldots, t_{T'/2}^*)$. Output $\mathsf{P}^* = \mathsf{iO}(G_1)$ where $G_1 = G_1[t_1^*, \ldots, t_{T'/2}^*, m_b, m_0, \alpha, K_{\mathsf{PPRF}}]$.

Note that the only difference between $\mathsf{Hybrid}_{1,Q}$ and $\mathsf{Hybrid}_{2,0}$ is how P^* is generated. In $\mathsf{Hybrid}_{1,Q}$, it is generated by obfuscating program $G[t_1^*, \ldots, t_{T'/2}^*, m_b, K_{\mathsf{PPRF}}]$, where is $\mathsf{Hybrid}_{2,0}$ it it generated by obfuscating program $G_1[t_1^*, \ldots, t_{T'/2}^*, m_b, m_0, 0, K_{\mathsf{PPRF}}]$. These programs have identical input output behavior. Thus if there exists an adversary $\mathcal{A}$ that distinguishes these hybrids with probability p, then, we can build a reduction that distinguishes iO with probability p. The reduction needs to invoke the code of $\mathcal{A}$, and answer polynomially many CCAVal queries. Therefore, its time is polynomial in time of $\mathcal{A}$ and the time of CCAVal. We set λ_{iO} large enough to ensure the following claim.

Lemma 2. *Assuming that* iO *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{\mathsf{iO}}})$*, then, for any adversary* $\mathcal{A}$ *of size polynomial in* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}_{1,Q}) = 1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}_{2,\alpha}) = 1]| \leq 2^{-\lambda_{\mathsf{iO}}}$$

Hybrid_3 : This hybrid is the same as the previous hybrid except to generate P^*, we obfuscate the circuit in Figure 5 by committing to m_0.

Note that the only difference between $\mathsf{Hybrid}_{2,2^{\ell_{h_{key}}}}$ and Hybrid_3 is how P^* is generated. In $\mathsf{Hybrid}_{2,2^{\ell_{h_{key}}}}$, it is generated by obfuscating program $G_1[t_1^*, \ldots, t_{T'/2}^*, m_b, m_0, \alpha = 2^{\ell_{h_{key}}}, K_{\mathsf{PPRF}}]$, where as in Hybrid_3 it it generated by obfuscating program $G[t_1^*, \ldots, t_{T'/2}^*, m_0, K_{\mathsf{PPRF}}]$. These programs have identical input output behavior. Thus if there exists an adversary $\mathcal{A}$ that distinguishes these hybrids with probability p, then, we can build a reduction that distinguishes iO with probability p. The reduction needs to invoke the code of $\mathcal{A}$, and answer polynomially many CCAVal queries. Therefore, its time is polynomial in time of $\mathcal{A}$ and the time of CCAVal. We set λ_{iO} large enough to ensure the following claim.

Lemma 3. *Assuming that* iO *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{iO}})$*, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}_3) = 1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}_{2,2^{\ell_{h_{key}}}}) = 1]| \leq 2^{-\lambda_{\mathsf{iO}}}$$

Indistinguishability between $\mathsf{Hybrid}_{2,\alpha}$ and $\mathsf{Hybrid}_{2,\alpha+1}$ To prove indistinguishability between $\mathsf{Hybrid}_{2,\alpha}$ and $\mathsf{Hybrid}_{2,\alpha+1}$ we introduce indistinguishable intermediate hybrids.
Hybrid'_0 : This hybrid is the same as $\mathsf{Hybrid}_{2,\alpha}$.

Hybrid'_1 : This hybrid is the same as $\mathsf{Hybrid}_{2,\alpha}$ except that we generate P^* differently. We puncture the PPRF key K_{PPRF} at α, and hardwire the response at $\rho = \alpha$. Namely, compute the punctured key k^*_{PPRF} at α. We compute a value v as follows. Compute $(r_1, r_2, r_3) \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \alpha)$. Then:

- Compute $c_0 = \mathsf{NICom}(0^{\ell_{h_{inp}}}; r_1)$ and for $i \in [T'/2]$, compute $c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t_i^*, m_b; r_{2,i})$,
- Compute $\pi = \mathsf{NIWI.P}(X, W; r_3)$ where $X = (c_0, c_1, \ldots, c_{T'/2})$ and $W = (m_b, \{r_{2,i}\}_{i \in [T']})$ are so that $(X, W) \in L_G$.
- Set $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$.

The Circuit $G_2[t_1, \ldots, t_{T'/2}, m_b, m_0, \alpha, K_{\mathsf{PPRF}}, v]$

Hardwired: Tags $(t_1, \ldots, t_{T'/2}) \in [T']^{T'/2}$, Messages m_b and m_0, PPRF key K_{PPRF}, $\alpha \in [0, 2^{\ell_{h_{key}}}]$ and a value v.

Input: $\rho \in \{0,1\}^{\ell_{h_{key}}}$

Computation: The computation can be divided into two cases.
Case: $\rho < \alpha$

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \rho)$ and parse $r = (r_1, r_{2,1}, \ldots, r_{2,T'/2}, r_3)$.
2. Compute $c_0 = \mathsf{NICom}(0^{\ell_{h_{inp}}}; r_1)$ and for $i \in [T'/2]$, compute $c_i = \mathsf{aCCA'.CCACommit}(t_i, m_0; r_{2,i})$,
3. Compute $\pi = \mathsf{NIWI.P}(X, W; r_3)$ where $X = (c_0, c_1, \ldots, c_{T'/2})$ and $W = (m_0, \{r_{2,i}\}_{i \in [T']})$ are so that $(X, W) \in L_G$ for the language L_G defined below.
4. Output $(c_0, c_1, \ldots, c_{T'/2}, \pi)$.

Case: $\rho > \alpha$

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \rho)$ and parse $r = (r_1, r_{2,1}, \ldots, r_{2,T'/2}, r_3)$.
2. Compute $c_0 = \mathsf{NICom}(0^{\ell_{h_{inp}}}; r_1)$ and for $i \in [T'/2]$, compute $c_i = \mathsf{aCCA'.CCACommit}(t_i, m_b; r_{2,i})$,
3. Compute $\pi = \mathsf{NIWI.P}(X, W; r_3)$ where $X = (c_0, c_1, \ldots, c_{T'/2})$ and $W = (m_b, \{r_{2,i}\}_{i \in [T']})$ are so that $(X, W) \in L_G$ for the language L_G defined below.
4. Output $(c_0, c_1, \ldots, c_{T'/2}, \pi)$.

Case: $\rho = \alpha$ Output v.
Language $L_G = L_{G_1} \vee L_{G,2}$:

$$L_{G,1} = \{(c_0, c_1, \ldots, c_{T'/2}) \mid \exists x \neq x' \in \{0,1\}^{\ell_{h_{in}}} \ s.t. \ \mathcal{H}(\rho, x) = \mathcal{H}(\rho, x') \wedge c_0 = \mathsf{NICom}(x, x')\}$$
$$L_{G,2} = \{(c_0, c_1, \ldots, c_{T'/2}) \mid \exists m \ s.t. \ \forall i \in [T'/2], \ c_i = \mathsf{aCCA'.CCACommit}(t_i, m)\}$$

Figure 7: The Circuit $G_2[t_1, \ldots, t_{T'/2}, m_b, m_0, \alpha, k_{\mathsf{PPRF}}, v]$

Output $\mathsf{P}^* = \mathsf{iO}(G_2)$ where $G_2 = G_2[t^*_1, \ldots, t^*_{T'/2}, m_b, m_0, \alpha, K^*_{\mathsf{PPRF}}, v]$ as described in Figure 7.

Note that the only difference between Hybrid'_0 and Hybrid'_1 is how P^* is generated. In Hybrid'_0, it is generated by obfuscating program $G_1[t^*_1, \ldots, t^*_{T'/2}, m_b, m_0, \alpha, K_{\mathsf{PPRF}}]$, where as in Hybrid'_1 it is generated by obfuscating $G_2[t^*_1, \ldots, t^*_{T'/2}, m_b, m_0, \alpha, K^*_{\mathsf{PPRF}}, v]$ where the key K^*_{PPRF} is punctured at α. Note that if PPRF key is correct at unpunctured points, these circuits have identical behavior on all inputs $\rho \neq \alpha$. On input $\rho = \alpha$, the outputs are made to be identical by setting $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$ which is computed as described in the Hybrid'_1 description. Thus, the security follows from the security of iO. We have that:

Lemma 4. *Assuming that* iO *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{iO}})$ *and* PPRF *is correct at unpunctured points, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}_3) = 1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}_{2,2^{\ell_{h_{key}}}}) = 1]| \leq 2^{-\lambda_{\mathsf{iO}}}$$

Hybrid'_2 : This hybrid is the same as the previous hybrid except while computing the hardwired value v, we replace $r_1, r_2, r_3 \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \alpha)$ to a truly random string.

Thus if there exists an adversary $\mathcal{A}$ that distinguishes these hybrids with probability p, then, we can build a reduction that breaks the pseudorandomness at the punctured points property of PPRF with probability p. The reduction needs to invoke the code of $\mathcal{A}$, and answer polynomially many CCAVal queries. Therefore, its time is polynomial in time of $\mathcal{A}$ and the time of CCAVal. We set λ_{PPRF} large enough to ensure the following claim.

Note that the only difference between Hybrid'_1 and Hybrid'_2 is how $r = (r_1, r_2, r_3)$ is generated. In Hybrid'_1 it is generated by computing $\mathsf{PPRF}(K_{\mathsf{PPRF}}, \alpha)$ where as in Hybrid'_2 it is generated r is sampled randomly. Note the in both the hybrids, the key appears in a punctured form K^*_{PPRF}, punctured at α. Thus if there exists an adversary $\mathcal{A}$ that distinguishes these hybrids with probability p, then, we can build a reduction that breaks the pseudorandomness at the punctured points property of PPRF with probability p. The reduction needs to invoke the code of $\mathcal{A}$, and answer polynomially many CCAVal queries. Therefore, its time is polynomial in time of $\mathcal{A}$ and the time of CCAVal. We set λ_{PPRF} large enough to ensure the following claim.

Lemma 5. *Assuming that* PPRF *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{\mathsf{PPRF}}})$*, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}'_1) = 1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}'_2) = 1]| \leq 2^{-\lambda_{\mathsf{PPRF}}}$$

Hybrid'_3 : This hybrid is the same as the previous hybrid except that while we compute the hardwired value $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$, we switch the commitment as $c_0 = \mathsf{NICom}(x_\alpha, x'_\alpha)$ where $(x_\alpha, x'_\alpha) \in (\{0,1\}^{\ell_{h_{in}}})^2$, $x_\alpha \neq x'_\alpha$ and $\mathcal{H}(\alpha, x_\alpha) = \mathcal{H}(\alpha, x_\alpha)$.

Note that the only difference between Hybrid'_2 and Hybrid'_3 is how hardwiring $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$ is generated. In particular, it is about how c_0 is generated. In Hybrid'_2 it is generated by computing c_0 as an honest commitment of $0^{2\ell_{h_{in}}}$ where as in Hybrid'_3 it is generated by committing to a collision x_α, x'_α for the hash key α. The openings for these commitments are not used to compute π. Thus if there exists an adversary $\mathcal{A}$ that distinguishes these hybrids with probability p, then, we can build a reduction that breaks the security of the commitment with probability p. The reduction is non uniform and must know a collision for hash key α. The reduction also needs to answer polynomially many CCAVal queries. Therefore, its time is polynomial in time of $\mathcal{A}$ and the time of CCAVal. We set λ_{NICom} large enough to ensure the following claim.

Lemma 6. *Assuming that* NICom *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{\mathsf{NICom}}})$*, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}'_2)=1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}'_3)=1]| \leq 2^{-\lambda_{\mathsf{NICom}}}$$

Hybrid'_4 : This hybrid is the same as the previous hybrid except that while we compute the hardwired value $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$, we replace π as $\pi = \mathsf{NIWI.P}(X, W)$ where $X = (c_0, \ldots, c_{T'/2})$ and W is consists of opening of $c_0 = \mathsf{NICom}(x_\alpha, x'_\alpha)$.

Note that the only difference between Hybrid'_3 and Hybrid'_4 is how hardwiring $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$ is generated. In particular, it is about how π is generated. In Hybrid'_3 it is generated by using openings of $c_1, \ldots, c_{T'/2}$ where as in Hybrid'_4 it is generated by using opening of c_0 as the witness. Thus if there exists an adversary $\mathcal{A}$ that distinguishes these hybrids with probability p, then, we can build a reduction that breaks the security of the NIWI with probability p. The reduction also needs to answer polynomially many CCAVal queries. Therefore, its time is polynomial in time of $\mathcal{A}$ and the time of CCAVal. We set λ_{NIWI} large enough to ensure the following claim.

Lemma 7. *Assuming that* NIWI *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{\mathsf{NIWI}}})$*, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}'_3)=1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}'_4)=1]| \leq 2^{-\lambda_{\mathsf{NIWI}}}$$

Hybrid'_5 : This hybrid is the same as the previous hybrid except that while we compute the hardwired value $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$, we replace for $i \in [T'/2]$, $c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t^*_i, m_0)$.

Note that the only difference between Hybrid'_4 and Hybrid'_5 is how hardwiring $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$ is generated. In particular, it is about how $c_1, \ldots, c_{T'/2}$ is generated. In Hybrid'_4 it is generated by computing each $c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t^*_i, m_b)$ for $i \in [T^*]$, where as in Hybrid'_4 it is generated by computing each $c_i = \mathsf{aCCA}'.\mathsf{CCACommit}(t^*_i, m_0)$ for $i \in [T^*]$. The openings of these commitments are not used in generating π. Note that in these hybrids, the adversary gets an oracle to $\mathsf{aCCA}'.\mathsf{CCAVal}()$ oracle but it does not query it on $(t^*_1, \ldots, t^*_{T'/2})$. Thus if there exists an adversary $\mathcal{A}$ that distinguishes these hybrids with probability p, then, we can build a reduction that breaks the security of the aCCA' with probability p.

Lemma 8. *Assuming that* aCCA' *is secure against circuits in* $\mathcal{C}_{\mathsf{aCCA}'}$*, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}'})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}'_4)=1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}'_5)=1]| \leq T' \cdot \epsilon_{\mathsf{aCCA}'}$$

Hybrid'_6 : This hybrid is the same as the previous hybrid except that while we compute the hardwired value $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$, we replace π as $\pi = \mathsf{NIWI.P}(X, W)$ where $X = (c_0, \ldots, c_{T'/2})$ and W is consists of opening of $c_1, \ldots, c_{T'/2}$ committing to m_0.

Hybrid'_5 and Hybrid'_6 are indistinguishable due to the security of NIWI, and follow similarly as in the indistinguishability between Hybrid'_3 and Hybrid'_4. Thus we have:

Lemma 9. *Assuming that* NIWI *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{\mathsf{NIWI}}})$*, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}'_5)=1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}'_6)=1]| \leq 2^{-\lambda_{\mathsf{NIWI}}}$$

Hybrid'_7 : This hybrid is the same as the previous hybrid except that while we compute the hardwired value $v = (c_0, c_1, \ldots, c_{T'/2}, \pi)$, we switch the commitment c_0 as $c_0 = \mathsf{NICom}(0^{2\ell_{h_{in}}})$.

Hybrid'_6 and Hybrid'_7 are indistinguishable due to the security of NICom, and follow similarly as in the indistinguishability between Hybrid'_2 and Hybrid'_3. Thus we have:

Lemma 10. *Assuming that* NICom *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{\mathsf{NICom}}})$*, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}'_6) = 1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}'_7) = 1]| \leq 2^{-\lambda_{\mathsf{NICom}}}$$

Hybrid'_8 : This hybrid is the same as the previous hybrid except while computing the hardwired value v, we replace r_1, r_2, r_3 from being truly random to be generated by $(r_1, r_2, r_3) \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \alpha)$.

Hybrid'_7 and Hybrid'_8 are indistinguishable due to the security of PPRF, and follow similarly as in the indistinguishability between Hybrid'_1 and Hybrid'_2. Thus we have:

Lemma 11. *Assuming that* PPRF *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{\mathsf{PPRF}}})$*, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}'_7) = 1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}'_8) = 1]| \leq 2^{-\lambda_{\mathsf{PPRF}}}$$

Hybrid'_9 : This hybrid is the same as $\mathsf{Hybrid}_{2,\alpha+1}$.

Hybrid'_8 and Hybrid'_9 are indistinguishable due to the security of iO, and follow similarly as in the indistinguishability between Hybrid'_0 and Hybrid'_1. Thus we have:

Lemma 12. *Assuming that* iO *is secure against circuits that run in time* $\mathsf{poly}(2^{\lambda_{\mathsf{iO}}})$ *and* PPRF *is correct at unpunctured points, then, for any adversary* $\mathcal{A}$ *of size* $\mathsf{poly}(s_{\mathsf{aCCA}})$*, it holds that:*

$$|\Pr[\mathcal{A}(\mathsf{Hybrid}'_8) = 1] - |\Pr[\mathcal{A}(\mathsf{Hybrid}'_9) = 1]| \leq 2^{-\lambda_{\mathsf{PPRF}}}$$

Final Advantage. Summing up the advantage, we have that the total advantage is bounded by:

$$2^{\ell_{h_{key}}} O(2^{-\lambda_{\mathsf{iO}}} + 2^{-\lambda_{\mathsf{NIWI}}} + 2^{-\lambda_{\mathsf{NICom}}} + 2^{-\lambda_{\mathsf{PPRF}}}) + 2^{\ell_{h_{key}}} O(T' \cdot \epsilon_{\mathsf{aCCA}'}) + O(2^{-\lambda_h}) \tag{3}$$

$$\leq \epsilon_{\mathsf{aCCA}} \tag{4}$$

The last inequality follows from the parameters we set.

7.3 Removing One-Tag Restriction

To remove one tag restriction, [Khu21] suggested the following approach. We explain the idea with the help of an ideal one-message zero-knowledge and standard one-round CCA commitments. Let nmc' be a commitment with tag space $T'(\lambda) = \underbrace{\log \ldots \log \lambda}_{O(1) \text{ times}}$ with one-tag restriction. We can build a CCA scheme without this restriction as follows. Suppose we want to commit to a message m with respect to a tag $t \in [T']$, then we can output the new commitment $\mathsf{nmc.CCACommit}(t, m)$ as: $(c_1, \ldots, c_{T'}, \pi)$ where:

- For $i \neq t$, $c_i = \mathsf{nmc.CCACommit}(i, m)$ is a commitment of m with tag i,

- $c_t = \bot$,
- π is proof that the commitment is generated in the way described above.

The reason this gets around the issue of one-tag restriction is because for any tag $t \neq t'$, we can run $\mathsf{nmc.CCAVal}(t', \star)$, by accessing just $\mathsf{nmc'.CCAVal}(t, \star)$. This is because in the new commitment to the message m, $\mathsf{nmc.CCACommit}(t, m)$ does not invoke $\mathsf{nmc'.CCACommit}()$ with respect to tag t (but uses every other tag), where as $\mathsf{nmc.CCACommit}(t', \star)$ will always have a component generated by using $\mathsf{nmc'.CCACommit}(t, \star)$ as $t' \neq t$. Further, the soundness of π will ensure that all commitments that are queried are consistently generated as in the procedure so that extraction using $\mathsf{nmc'.CCAVal}(t, \star)$ is correct. The security can then be proven by first simulating π and then switching the commitments one by one.

While this is the idea relying on a one-message zero-knowledge, the above idea can be formalized without such a zero-knowledge relying on receiver assistance. Let $\mathsf{aCCA'}$ be the underlying receiver assisted CCA scheme for tag space $[T']$ above. We build our scheme aCCA without one-tag restriction for the same tag $[T']$ following the same approach as our tag amplification transformation, except that the obfuscation corresponds to a slightly different program. The only change is that now, on input the receiver string τ it will produce $c_0, c_1, \ldots, c_{T'}, \pi$ where $c_1, \ldots, c_{T'}$ are generated in the way described above.

We now describe this transformation below. We use the same primitives, notation and parameters as in our tag amplification transformation, the only change being that $\mathsf{aCCA'}$ suffers from one-tag restriction and that $T = T'$. The proof of security is essentially is essentially the same as in our tag amplification construction. We defer it to the full version.

$\mathsf{aCCA.CCACommit}(\mathsf{tag}, m; r)$: Compute the following steps.

- Sample a PPRF key $K_{\mathsf{PPRF}} \leftarrow \{0,1\}^{\ell_{\mathsf{PPRF}}}$,
- Compute $\widetilde{F} \leftarrow \mathsf{iO}(F[\mathsf{tag}, m, K_{\mathsf{PPRF}}])$ by obfuscating the circuit described in Figure 8. Output $\widetilde{F}$.

$\mathsf{aCCA.Opening}(\tau, \mathsf{tag}, \widetilde{G}, m, r)$: Compute the following steps.

- Parse $\tau = (\rho, \rho')$ where $\rho \in \{0,1\}^{\ell_{h_{key}}}$ and $\rho' \in \{0,1\}^{\ell_{\mathsf{aCCA'}}}$,
- Check if $\widetilde{F} = \mathsf{aCCA.CCACommit}(\mathsf{tag}, m; r)$. Abort if its not the case. Derive the PPRF key K_{PPRF} used in code of F described in Figure 8.
- Compute $\widetilde{F}[\rho] = (c_0, c_1, \ldots, c_{T'}, \pi)$,
- From the code of Figure 8, use the PPRF key k_{PPRF} to derive r'_t as in the code so that $c_t = \mathsf{aCCA'.CCACommit}(t, m; r'_i)$ for $t \in [T'] \setminus \mathsf{tag}$. Compute and output $\sigma_t = \mathsf{aCCA'.Opening}(\rho', t, c_t, m, r'_t)$ for $i \in [T'] \setminus t$.

$\mathsf{aCCA.Open}(\tau, \mathsf{tag}, \widetilde{F}, m, \sigma)$: Compute the following steps.

- Parse $\tau = (\rho, \rho')$ where $\rho \in \{0,1\}^{\ell_{h_{key}}}$ and $\rho' \in \{0,1\}^{\ell_{\mathsf{aCCA'}}}$,
- Compute $\widetilde{F}[\rho] = (c_0, c_1, \ldots, c_{T'}, \pi)$ and verify π using $\mathsf{NIWI.Vf}$ for the language described in 8. Abort if the proof does not verify,
- Output 1 if for every $t \in [T'] \setminus \mathsf{tag}$, $\mathsf{aCCA'.Open}(\rho', t, c_t, m, \sigma_t)$. Output $\bot$ otherwise.

Remark 3 (Opening algrorithm for base scheme with T' tags)**.** *For base commitments as in [BL18, LPS20], the* $\mathsf{aCCA'.Opening}$ *simply outputs the randomness to commit the message.*

The Circuit $F[\mathsf{tag}, m, K_{\mathsf{PPRF}}]$

Hardwired: Tag $\mathsf{tag} \in [T']$, Message m and PPRF key K_{PPRF},

Input: $\rho \in \{0,1\}^{\ell_{h_{key}}}$

Computation:

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K_{\mathsf{PPRF}}, \rho)$ and parse $r = (r_1, r_{2,1}, \ldots, r_{2,T'}, r_3)$. Compute:
   - $c_0 = \mathsf{NICom}(0^{\ell_{h_{inp}}}; r_1)$,
   - For $t \in [T']$, and $t \neq \mathsf{tag}$, compute $c_t = \mathsf{aCCA'.CCACommit}(t, m; r_{2,i})$,
   - Set $c_{\mathsf{tag}} = \bot$,
   - Compute $\pi = \mathsf{NIWI.P}(X, W; r_3)$ where $X = (c_0, c_1, \ldots, c_{T'})$ and $W = (m, \{r_{2,i}\}_{i \in [T']})$ are so that $(X, W) \in L_F$ for the language L_F defined below.
2. Output $(c_0, c_1, \ldots, c_{T'}, \pi)$.

Language $L_G = L_{F_1} \vee L_{F,2}$:

$$L_{F,1} = \{(c_0, c_1, \ldots, c_{T'}) \mid \exists x \neq x' \in \{0,1\}^{\ell_{h_{in}}} \ s.t.\ \mathcal{H}(\rho, x) = \mathcal{H}(\rho, x') \wedge c_0 = \mathsf{NICom}(x, x')\}$$
$$L_{F,2} = \{(c_0, c_1, \ldots, c_{T'}) \mid \exists m \ s.t.\ \forall t \in [T'] \setminus \mathsf{tag},\ c_t = \mathsf{aCCA'.CCACommit}(t, m) \wedge c_{\mathsf{tag}} = \bot\}$$

Figure 8: The Circuit $F[\mathsf{tag}, m, k_{\mathsf{PPRF}}]$

8 Primitives used for Constructing Our Zero-Knowledge Protocol

In this section, we define and construct two tools that we will use to build our reusable statistical ZK arguments with sometimes statistical soundness. We first discuss their definitions and then show how to construct them. The first notion is that of Non-interactive Distributional Indistinguishability NIDI [Khu21]. Unfortunately, we can't directly use their construction and therefore we provide the construction that satisfies our requirements. The second notion is that of a sometimes extractable equivocal commitments SEE, which is a contribution of this work. A reader may skip this section and proceed onto Section 2.2.1 for an overview of our zero-knowledge argument construction, and then come back here for formal details about these ingredients.

8.1 Non-Interactive Distributional Indistinguishability

In this section, we define the notion of Non-Interactive Distributional Indistinguishability arguments (NIDI for short). The definitions below are strengthenings of analogous definitions given by Khurana [Khu21], where the difference is that their definitions assume that the verifier's message comes after the prover's message; see Remark 5 for details.

Definition 18 (Syntax of NIDI)**.** *A* NIDI *for an* NP *language L and its relation R_L consists of the following algorithms.*

- $\mathsf{P}(1^{\lambda_S}, 1^{\lambda_D}, \mathcal{D})$: *The prove algorithm takes as input two security parameters 1^{λ_S} and 1^{λ_D} (one for the soundness property, and one for the distribution indistinguishability property), a polynomial time sampler $\mathcal{D}(\cdot)$ that on input λ_D samples from $(\mathcal{X}, \mathcal{W})$ consisting of tuples that are in R_L. It outputs a proof string π.*

- $\mathsf{V}(\tau, \pi)$: *The verification algorithm is a deterministic polynomial time algorithm that takes as input a string* $\tau \in \{0,1\}^{\ell_{\mathsf{NIDI}}(\lambda_S)}$ *for some polynomial* ℓ_{NIDI}*, a proof* π*, and it outputs a string in* $x \in \perp \cup \{0,1\}^*$.

A NIDI scheme satisfies a number of different properties: completeness, soundness and distributional indistinguishability.

Definition 19 (Completeness)**.** *We require that for any poly-time samplable distribution* $\mathcal{D} = (\mathcal{X}, \mathcal{W})$ *supported over instance-witness pairs in* R_L*, we have that for every* $\lambda_S, \lambda_D \in \mathbb{N}$*:*

$$\Pr_{\tau,\pi}[x \in L \mid \mathsf{V}(\tau, \pi) = x] = 1,$$

where $\pi \leftarrow \mathsf{P}(1^{\lambda_S}, 1^{\lambda_D}, \mathcal{D})$ *and* $\tau \leftarrow \{0,1\}^{\ell_{\mathsf{NIDI}}(\lambda_S)}$.

Definition 20 ($(\mathcal{C}_D, \mathcal{C}_{DI}, \epsilon_D, \epsilon_{DI})$- Distributional Indistinguishability)**.** *Let* $\mathcal{D}_0 = (\mathcal{X}_0, \mathcal{W}_0)$ *and* $\mathcal{D}_1 = (\mathcal{X}_1, \mathcal{W}_1)$ *be two polynomial-time distribution samplers supported over tuples in* R_L*. Further, assume that* $\mathcal{X}_0$ *and* $\mathcal{X}_1$ *are* $(\mathcal{C}_D, \epsilon_D)$ *indistinguishable. Then, we require that:*

$$\mathsf{P}(1^{\lambda_S}, 1^{\lambda_D}, \mathcal{D}_0) \approx_{\mathcal{C}_{DI}, \epsilon_{DI}} \mathsf{P}(1^{\lambda_S}, 1^{\lambda_D}, \mathcal{D}_1).$$

Definition 21 (Completeness, Extraction)**.** *There exist a (possibly inefficient) algorithm* $E : \{0,1\}^* \to \{0,1\}$ *with the following properties. Let* $\lambda_S, \lambda_D \in \mathbb{N}$*,* $\tau \in \{0,1\}^{\ell_{\mathsf{NIDI}}(\lambda_S)}$ *and* π *be any proof string such that* $\mathsf{V}(\tau, \pi) \to x$ *where* $x \neq \perp$*. Then:*

- $E(\tau, \pi) = 1 \implies x \in L$.
- *For any polynomial time samplable distribution* $\mathcal{D} = (\mathcal{X}, \mathcal{W})$ *supported over tuples in* R_L*, it holds that:*

$$\Pr\left[E(\tau, \pi) = 1 \;\middle|\; \begin{array}{l} \tau \xleftarrow{\$} \{0,1\}^{\ell_{\mathsf{NIDI}}(\lambda_S)} \\ \pi \leftarrow \mathsf{P}(1^{\lambda_S}, 1^{\lambda_D}, \mathcal{D}) \end{array}\right] = 1.$$

Definition 22 ($(\mathcal{C}_S, \epsilon_S)$-Soundness)**.** *We define the following security game played between the adversary* $\mathcal{A} \in \mathcal{C}_S$ *and the challenger. We denote it by* $\mathsf{expt}_{\mathcal{A},\mathsf{NIDI},sound}(1^{\lambda_S}, 1^{\lambda_D})$*:*

1. $\mathcal{A}$ *is given* $1^{\lambda_S}, 1^{\lambda_D}$ *as the input.*
2. *The challenger manages a list* List *that is initially empty. The contents of the list are visible to the adversary at all stages.*
3. *Adversary can ask adaptively a polynomial number of* τ*-query. If that happens, sample* $\tau' \leftarrow \{0,1\}^{\ell_{\mathsf{NIDI}}(\lambda_S)}$ *and append* τ' *to* List.
4. *Adversary outputs a proof string* π *and a* $\tau \in$ List*. The adversary wins if* $\mathsf{V}(\tau, \pi) = x$ *where* $x \neq \perp$ *and* $E(\tau, \pi) = 0$.

The NIDI *scheme satisfies* $(\mathcal{C}_S, \epsilon_S)$*-soundness if for all adversaries* $\mathcal{A} \in \mathcal{C}_S$*:*

$$\Pr[\mathsf{expt}_{\mathcal{A},\mathsf{NIDI},sound}(1^{\lambda_S}, 1^{\lambda_D}) = 1] \leq \epsilon_S$$

Remark 4. *Observe that the last two properties gives rise to a meaningful soundness property. The extraction property (Definition 21) ensures that whenever* $x \notin L$*, if* $\mathsf{V}(\tau, \pi) = x$ *then* $E(\tau, \pi) \neq 1$*. The Soundness property (Definition 22) then says that for a computationally bounded adversary it is hard to come up with a proof string* π *such that* $\mathsf{V}(\tau, \pi) = x$ *and* $E(\tau, \pi) \neq 1$*. This rules out a computationally bounded adversary producing false instances.*

Remark 5 (weaker soundness requirement)**.** *One could ask for a weaker soundness requirement where the proof string is required to be published before making any* τ *query. Such a* NIDI *will not be sufficient for us. The protocol in [Khu21] satisfies this weaker property.*

8.2 Sometimes Extractable Equivocal Commitments

In this section we define the notion of sometimes extractable equivocal commitments SEE that we use. These commitments are inspired by the ones used to build statistical ZAP arguments [BFJ+20, GJJM20].

Definition 23. *An* SEE *is a tuple of three p.p.t. algorithms* $\mathsf{Com}_{1,R}, \mathsf{Com}_{1,C}, \mathsf{Com}_{2,C}$ *with the following syntax:*

- $\mathsf{Com}_{1,R}(1^\lambda, 1^t, 1^\mu, \mathbf{b_R}; r) \to \mathsf{com}_{1,R}$. *The* $\mathsf{Com}_{1,R}$ *denotes the first receiver message. It takes as input three security parameters* λ, t, μ *along with a string* $\mathbf{b_R} \in \{0,1\}^\ell$ *for some polynomial* $\ell = \ell(\mu)$*. It outputs* $\mathsf{com}_{1,R}$*.*
- $\mathsf{Com}_{1,C}(1^\lambda, 1^t, 1^\mu, \mathbf{b_C}) \to \mathsf{com}_{1,C}$. *The* $\mathsf{Com}_{1,C}$ *denotes the first committer message. It takes as input three security parameters* λ, t, μ *along with a string* $\mathbf{b_C} \in \{0,1\}^\ell$*. It deterministically outputs* $\mathsf{com}_{1,C}$*.*
- $\mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, m; r') \to \mathsf{com}_{2,C}$. *The* $\mathsf{Com}_{2,C}$ *denotes the second committer message. It takes as input first committer and receiver messages* $\mathsf{com}_{1,R}, \mathsf{com}_{1,C}$ *along with a message* m *and outputs* $\mathsf{com}_{2,C}$ *which is referred to as the commitment.*

Such a scheme satisfies the following properties.

$(\mathcal{C}_D, \epsilon_D)$-Indistinguishability of $\mathsf{Com}_{1,R}$. Let $\lambda \in \mathbb{N}$ and $\mu \in \lambda^{O(1)}$, $t \in \lambda^{\Omega(1)(\log\log\lambda)^{-1}} \cap \lambda^{O(1)}$ and $\mathbf{b} \in \{0,1\}^\ell$. Then, it holds that:

$$\mathsf{Com}_{1,R}(1^\lambda, 1^\mu, 1^t, \mathbf{b}) \approx_{\mathcal{C}_D, \epsilon_D} \mathsf{Com}_{1,R}(1^\lambda, 1^\mu, 1^t, 0^\ell).$$

Verifiability of $\mathsf{Com}_{1,C}$. There exists a deterministic polynomial time algorithm Vf that takes as input $1^\lambda, 1^t, 1^\mu$ and $\mathsf{com}_{1,C}$ and outputs 1 if and only if $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(1^\lambda, 1^t, 1^\mu, \mathbf{b})$ for some $\mathbf{b} \in \{0,1\}^\ell$.

Extraction when $\mathbf{b}_R = \mathbf{b}_C$ There exist a deterministic polynomial time algorithm Dec such that the following holds. Let $\lambda \in \mathbb{N}$, $\mu = \lambda^{O(1)}$, $t \in \lambda^{\Omega(1)(\log\log\lambda)^{-1}} \cap \lambda^{O(1)}$. Then, for any $\mathbf{b} \leftarrow \{0,1\}^\ell$ and any message $m \in \{0,1\}^*$

$$\Pr_{r,r'}[\mathsf{Dec}(\mathbf{b}, r, \mathsf{com}_{1,C}, \mathsf{com}_{1,R}, \mathsf{com}_{2,C}) = m] = 1,$$

where, $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(1^\lambda, 1^\mu, 1^t, \mathbf{b})$, $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(1^\lambda, 1^\mu, 1^t, \mathbf{b}; r)$ and $\mathsf{com}_{2,C} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, m; r')$.

Equivocation when $\mathbf{b}_R \neq \mathbf{b}_C$. We require that these exist an algorithm $\mathcal{S}$ such that the following holds. Let $\lambda \in \mathbb{N}$, $\mu = \lambda^{\Theta(1)}$ and $t = \lambda^{\Omega(1)(\log\log\lambda)^{-1}} \cap \lambda^{O(1)}$. Let $\mathbf{b}_1 \neq \mathbf{b}_2$ be both in $\{0,1\}^\ell$. Then, for any $m \in \{0,1\}^*$, with probability 1 over the coins of $\mathsf{Com}_{1,R} = \mathsf{Com}_1(1^\lambda, 1^\mu, 1^t, \mathbf{b}_1)$ and $\mathsf{Com}_{1,C}(1^\lambda, 1^\mu, 1^t, \mathbf{b}_2)$, the following distributions are identical:

- Distribution 1: $\mathsf{com}_{2,C} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, m; r)$. Output $(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathsf{com}_{2,C}, m, r)$.
- Distribution 2: $\mathsf{com}_{2,C} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, 0^{|m|}; r')$. Compute $\mathcal{S}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, r', m) \to r$. Output $(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathsf{com}_{2,C}, m, r)$.

Additionally, $\mathcal{S}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, r', m)$ runs in time $2^t \cdot \mathsf{poly}(\lambda, |m|)$.

Hard to force $\mathbf{b}_R = \mathbf{b}_C$ by adversaries in $\mathcal{C}_{\mathcal{A}}$. Let $\lambda \in \mathbb{N}$, $\mu = \lambda^{\Theta(1)}$ and $t = \lambda^{\Omega(1)(\log\log\lambda)^{-1}} \cap \lambda^{O(1)}$. Then, for any adversary $\mathcal{A}$ in class $\mathcal{C}_{\mathcal{A}}$, the advantage of any adversary in the following experiment is $2^{-\mu}$.

- The challenger samples $\mathbf{b}_C \leftarrow \{0,1\}^\ell$ and sends $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(1^\lambda, 1^\mu, 1^t, \mathbf{b}_C)$.
- Adversary sends out $\mathsf{com}_{1,R}$. Adversary wins if it outputs $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(1^\lambda, 1^\mu, 1^t, \mathbf{b}_C; r)$ for some $r \in \{0,1\}^*$.

8.3 Construction of NIDI

We now describe our construction of the NIDI scheme (for any NP language L with its relation verifier R) satisfying all the properties described in Section 8.1. The scheme is almost identical to the construction of [Khu21] except for one change. We will highlight the change in the construction below in red. Before we proceed we describe the complexity classes involved.

Complexity Classes. We have the following:

- **Initial Distribution Properties.** We will consider distributions that are $\epsilon_D(\lambda_D) = 2^{-\lambda_D}$ indistinguishable against adversaries in the class $\mathcal{C}_D$ which consists of all circuits of depth $\mathsf{poly}(\lambda_D)$ and size 2^{λ_D}.
- **Properties of the resulting NIDI Proofs.** We will guarantee that the NIDI proofs for such distributions are indistinguishable for $\mathcal{C}_{DI} = \mathcal{C}_D$ described above (same circuit class). The advantage of adversaries in the security game will be bounded by $\epsilon_{DI} = O(\epsilon_D \cdot 2^{\ell_{\mathsf{NIDI}}(\lambda_S)})$.
- **Soundness properties.** We will ensure that the soundness holds against adversaries in $\mathcal{C}_S$ which consists of all adversaries of size 2^{λ_S}. The advantage will be bounded by $\epsilon_S = 2^{-\lambda_S}$.

Used Primitives. We make use of the following primitives and instantiated with the following parameters. These instantiated parameters for the primitives we use are loose for what we require.

OWP: We require a one way permutation OWP. We assume that OWP is secure against adversaries of size $2^{\lambda_{\mathsf{OWP}}}$, with advantage bounded by $2^{-\lambda_{\mathsf{OWP}}}$, where λ_{OWP} is the security parameter of the one-way permutation. Let the function be described as $\mathsf{OWP} : \{0,1\}^{\ell_{\mathsf{OWP}}} \to \{0,1\}^{\ell_{\mathsf{OWP}}}$ where $\ell_{\mathsf{OWP}} = \ell_{\mathsf{OWP}}(\lambda_{\mathsf{OWP}})$ is some polynomial in λ_{OWP}. We set $\lambda_{\mathsf{OWP}} = \lambda_S$ and $\ell = \ell_{\mathsf{OWP}}(\lambda_S)$. Such a function can be constructed assuming the subexponential time and advantage hardness of DDH/SXDH assumption.

Indistinguishability Obfuscation: We require an indistinguishability Obfuscator iO. This scheme uses λ_{iO} as the security parameter and is secure against adversaries of size $2^{\lambda_{\mathsf{iO}}}$ with advantage $2^{-\lambda_{\mathsf{iO}}}$. Such a primitive can be built using well-studied assumptions as shown in [JLS21a, JLS21b]. We set λ_{iO} as a large enough polynomial. In particular, setting $\lambda_{\mathsf{iO}} = \ell_{\mathsf{OWP}}\lambda_D$ suffices.

Time-Lock Puzzles: We require a time lock puzzle as in Definition 6. The TLP satisfies the following parameters.

- $\lambda_{\mathsf{TLP}} = \lambda_D \ell_{\mathsf{OWP}}$,
- $t_{\mathsf{TLP}} = \lambda_S^\rho$ for a small constant $\rho > 0$,
- The function $D(t) = 2^{t^\epsilon_{\mathsf{TLP}}}$ for some constant $\epsilon > 0$.

Therefore, TLP with these parameters ensures the security against adversary of size $2^{\lambda_{\mathsf{TLP}}}$ and depth bounded by $2^{t^{\epsilon}_{\mathsf{TLP}}}$ with the advantage bounded by $2^{\lambda_{\mathsf{TLP}}}$. Further, Solve can be run by a circuit of depth $\mathsf{poly}(2^{t_{\mathsf{TLP}}})$.

PUNCTURABLE PRF: We require a puncturable PRF, $\mathsf{PPRF} = (\mathsf{Puncture}, \mathsf{Eval})$. Assume the length of the key is randomly chosen of length $\ell_{\mathsf{PPRF}}(\lambda_{\mathsf{PPRF}})$ where λ_{PPRF} is its security parameter. The length of the output is some polynomial implicit in the scheme. We assume that the PPRF is secure against adversaries of size $2^{\lambda_{\mathsf{PPRF}}}$ with a maximum advantage of $2^{-\lambda_{\mathsf{PPRF}}}$. We set $\lambda_{\mathsf{PPRF}} = \lambda_D \cdot \ell$.

NIWI: We require a non-interactive witness indistinguishable proof NIWI for NP, that is secure against adversaries of size $2^{\lambda_{\mathsf{NIWI}}}$ with advantage bounded by $2^{-\lambda_{\mathsf{NIWI}}}$. We set $\lambda_{\mathsf{NIWI}} = \ell \cdot \lambda_D$. NIWIs can be instantiated assuming the subexponential time and advantage security of the SXDH assumption over bilinear maps.

Theorem 4. *Assume that the following assumptions hold:*

- *A subexponentially secure indistinguishability obfuscator exists,*
- *A time lock puzzle as in Definition 6 exist,*
- *SXDH is subexponentially secure against adversaries of subexponential size,*

then, there exist a NIDI *scheme that satisfies security definitions in Definitions 19, 22, 20, 21, and is secure against adversaries of subexponential size.*

The only difference from the primitives used in the construction by [Khu21] is the usage of a TLP as opposed to a public-key encryption scheme. This is the key component that helps us argue security in presence of adaptive τ queries.

Construction. We now describe the construction.

$\mathsf{NIDI.P}(1^{\lambda_S}, 1^{\lambda_D}, \mathcal{D})$: Sample a PPRF key $K \leftarrow \{0,1\}^{\ell_{\mathsf{PPRF}}}$.

The proving algorithm outputs $\widetilde{C} = \mathsf{iO}(C[\mathcal{D}, K])$ where the program $C[\mathcal{D}, K]$ is described in Figure 9.

$\mathsf{NIDI.V}(\tau, \widetilde{C})$: Run $\widetilde{C}(\tau)$. If this evaluation outputs $\perp$, output $\perp$. Otherwise, parse the output as (x, c, π). Run $\mathsf{NIWI.V}(x, c, \pi)$ for the language L'. If the verification fails output $\perp$. Otherwise, output x.

$E(\tau, \widetilde{C})$: Run $\widetilde{C}(\tau)$. Output 0 if this yields $\perp$. Otherwise parse the output as (x, c, π). Run $\mathsf{NIWI.V}(x, c, \pi)$. If the proof does not verify, output 0. Otherwise, check if $c = \mathsf{TLP.PGen}(\alpha)$ for some α. If this is not the case or $\mathsf{OWP}(\alpha) \neq \tau$, then output 1. Otherwise output 0.

Observe that the completeness property is immediate. Similarly the distributional indistinguishability property argument is also identical to the proof in [Khu21] because the public key encryption is replaced with a time-lock puzzle. All we need for the proof is for the component c to satisfy indistinguishability property. We still give a sketch of the proof in [Khu21] for completeness.

The Circuit $C[\mathcal{D}, K]$

Hardwired: The PPRF key K, and the distribution sampled $\mathcal{D}$.

Input: $\tau \in \{0,1\}^\ell$

Computation:

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K, \tau)$.
2. Parse $r = (r_1, r_2, r_3)$. Compute:
   - $(x, w) = \mathcal{D}(r_1)$,
   - $c = \mathsf{TLP.PGen}(0^\ell; r_2)$,
3. For the statement $(x, c) \in L'$, compute $\pi = \mathsf{NIWI.P}((x, c), w; r_3)$. We define the language

$$L' = \{(x', c') \mid \exists w' : R(x', w') = 1 \vee \exists \alpha : (c' = \mathsf{TLP.PGen}(\alpha) \wedge \mathsf{OWP}(\alpha) = \tau)\}$$

4. Output (x, c, π).

The code highlighted in red is the only difference from the construction proposed by [Khu21]. In their scheme, they generate $c = \mathsf{Enc}(\mathsf{pk}, 0^\ell)$ where pk is a public key for a dense cryptosystem, which is sampled and hardwired in the program. Any adversary breaking the soundness must commit/encrypt to an element in $\mathsf{OWP}^{-1}(\tau)$, and the reduction breaks open the encryption to win in the OWP game. This breaking is done by non-uniformly choosing the secret key for the public key pk. This only allows τ queries to come after the prover outputs a NIDI proof. A TLP helps us to bypass this issue.

Figure 9: The Circuit $C[\mathcal{D}, K]$

Sketch of Indistinguishability: The idea for indistinguishability is to go input by input as common in applications of iO. Consider two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ which yields instances that are $(\mathcal{C}_D, \epsilon_D)$ indistinguishable. The proof will follow the following strategy. We will define 2^ℓ hybrids where a typical hybrid ($\mathbf{Hybrid}_{\tau'}$) is indexed by $\tau' \in [2^\ell]$. In $\mathbf{Hybrid}_{\tau'}$, we will generate an obfuscation $\widetilde{C}$ of program $C[\mathcal{D}_0, \mathcal{D}_1, K, \tau']$ described in Figure 10. Now to prove indistinguishability, we need to prove that $\mathbf{Hybrid}_{\tau'}$ and $\mathbf{Hybrid}_{\tau'}$ are $O(2^{-\lambda_D})$ indistinguishable for circuits in $\mathcal{C}_D$. This will yield a total advantage of $O(2^{-\lambda_D}\ell 2^\ell)$. We can do this again by using standard tricks. Observe that the only change in the $C[\mathcal{D}_0, \mathcal{D}_1, K, \tau']$ and $C[\mathcal{D}_0, \mathcal{D}_1, K, \tau'+1]$ is its behavior at the input $\tau'+1$. In this case, we take the following hybrids. The indistinguishability between the hybrids are immediate and follow similarly to [Khu21].

- $\mathbf{Hybrid}'_0$: This is the same as $\mathbf{Hybrid}_{\tau'}$.
- $\mathbf{Hybrid}'_1$: In this hybrid the only is change is that, we puncture the PRF key K^* at $\tau'+1$ and use it to generate the circuit we obfuscate. To do so, we hardwire the output (x, c, π) generated from $\mathcal{D}_0$ as before using $(r_1, r_2, r_3) = \mathsf{PPRF.Eval}(K, \tau'+1)$. This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_{\mathsf{iO}}})$ due to the correctness property of the PPRF and the security of iO.
- $\mathbf{Hybrid}'_2$: In this hybrid the only change from the previous hybrid is that we generate (x, c, π) from $\mathcal{D}_0$ but now using true randomness (r_1, r_2, r_3). This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_{\mathsf{PPRF}}})$ due to the security property of the PPRF.
- $\mathbf{Hybrid}'_3$: In this hybrid the only change is that we generate (x, c, π), where c is computed as $\mathsf{TLP.PGen}(\alpha)$ where $\mathsf{OWP}(\alpha) = \tau'+1$. This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_{\mathsf{TLP}}})$ due to the security property of the TLP.
- $\mathbf{Hybrid}'_4$: In this hybrid, the only change is that we generate (x, c, π), by using opening of c as a witness to generate π. This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_{\mathsf{NIWI}}})$ due to the security property of the NIWI.
- $\mathbf{Hybrid}'_5$: In this hybrid the only change is that, we generate (x, c, π), by switching x to be sampled from $\mathcal{D}_1$. This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_D})$ due to the indistinguishability property of $\mathcal{D}_0$ and $\mathcal{D}_1$.
- $\mathbf{Hybrid}'_6$: In this hybrid the only change is that we generate (x, c, π), by using a witness of x to generate π. This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_{\mathsf{NIWI}}})$ due to the security property of NIWI.
- $\mathbf{Hybrid}'_7$: In this hybrid the only change is that we generate (x, c, π) where c is computed as $\mathsf{TLP.PGen}(0^\ell)$. This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_{\mathsf{TLP}}})$ due to the security property of TLP.
- $\mathbf{Hybrid}'_8$: In this hybrid the only change is that, we generate (x, c, π) by using $(r_1, r_2, r_3) = \mathsf{PPRF.Eval}(K, \tau'+1)$. This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_{\mathsf{PPRF}}})$ due to the security property of the PPRF.
- $\mathbf{Hybrid}'_9$: This hybrid is the same as $\mathbf{Hybrid}_{\tau'+1}$. This hybrid is indistinguishable to the previous hybrid against adversaries in $\mathcal{C}_D$ with advantage $O(2^{-\lambda_{\mathsf{iO}}})$ due to the correctness property of the PPRF and the security of iO.

Observe that the parameters $\lambda_{\mathsf{iO}}, \lambda_{\mathsf{NIWI}}, \lambda_{\mathsf{PPRF}}$ are set to be larger than $\lambda_D \ell$. Thus, the total advantage is bounded by $O(2^{-\lambda_D} + 2^{-\ell\lambda_D}) = O(2^{-\lambda_D})$. This finishes the overview.

The Circuit $C[\mathcal{D}_0, \mathcal{D}_1, K, \tau']$

Hardwired: The PPRF key K, and the distribution sampled $\mathcal{D}$.

Input: $\tau \in \{0,1\}^\ell$

Computation:

1. Compute $r \leftarrow \mathsf{PPRF.Eval}(K, \tau)$.
2. Parse $r = (r_1, r_2, r_3)$. Compute:
   - If $\tau \leq \tau'$, then $(x, w) = \mathcal{D}_1(r_1)$ otherwise $(x, w) = \mathcal{D}_0(r_1)$.
   - $c = \mathsf{TLP.PGen}(0^\ell; r_2)$,
3. For the statement $(x, c) \in L'$, compute $\pi = \mathsf{NIWI.P}((x, c), w; r_3)$. We define the language
$$L' = \{(x', c') \ | \exists w' : \ R(x', w') = 1 \vee \exists \alpha : \ (c' = \mathsf{TLP.PGen}(\alpha) \wedge \mathsf{OWP}(\alpha) = \tau)\}$$
4. Output (x, c, π).

Figure 10: The Circuit $C[\mathcal{D}_0, \mathcal{D}_1, K, \tau']$

We now focus on the soundness argument:

Sketch of Soundness. Consider a circuit $\mathcal{A}$ of size 2^{λ_S} in the soundness security game. Assume that the adversary wins in the soundness experiment with probability ϵ, then we will show that we can build a reduction of size $O(2^{\lambda_S})$ winning in the OWP inversion game with the ϵ/Q for some polynomial. Remember in the soundness game adversary is given a list $\tau_1, \ldots, \tau_Q$ of randomly chosen elements for some polynomial Q and it outputs $\widetilde{C}$ and an index $i \in [Q]$. For this $\widetilde{C}$, it holds that $\widetilde{C}[\tau_i] = (x_i, c_i, \pi_i)$ such that π_i verifies and $E(\tau_i, \widetilde{C}) = 0$. This means that c_i must be of the form $\mathsf{TLP.PGen}(\alpha_i)$ where $\mathsf{OWP}(\alpha_i) = \tau_i$. The reduction simply runs $\mathsf{TLP.Solve}(c_i)$ and outputs α_i as a preimage of τ_i. This means that the reduction succeeds with advantage at least ϵ/Q. Reduction needs to run $\mathcal{A}$ and then run $\mathsf{TLP.Solve}$, which runs in time polynomial in 2^{λ^ρ} for some small constant ρ. Thus, this takes $O(2^{\lambda_S})$ time as $\lambda_S = \lambda$.

8.4 Construction of Sometimes Extractable Equivocal Commitments

In this section, we present our construction of a sometimes extractable equivocal commitments. But first we specify the various class of adversary that we will handle in this scheme. Refer Definition 23 for these notations. Let λ, μ, t be three parameters involved where $\lambda \in \mathbb{N}$, $\mu = \lambda^{\Theta(1)}$ and $t \in \lambda^{\Omega(1)(\log\log\lambda)^{-1}}$.

Definition 24 (Complexity Parameters for SEE)**.** *Consider the following complexity classes as a function of λ, μ, t:*

- $\mathcal{C}_D$: *consists of all circuits of any polynomial depth and size polynomial in 2^λ.*
- ϵ_D : *is set to $2^{-\lambda}$.*

- $\mathcal{C}_{\mathcal{A}}$ *will be set to all circuits of size* 2^{μ}.

In order to build this primitive we make use of the following primitives:

Used Primitives. We make use of the following primitives and instantiated with the following parameters. These instantiated parameters for the primitives we use are loose for what we require.

One-Way Permutation: We require a one way permutation OWP. We assume that OWP is secure against adversaries of size polynomial in $2^{\lambda_{\mathsf{OWP}}}$, with advantage bounded by $2^{-\lambda_{\mathsf{OWP}}}$, where λ_{OWP} is the security parameter of the one-way permutation. Let the function be described as $\mathsf{OWP} : \{0,1\}^{\ell} \to \{0,1\}^{\ell}$ where $\ell = \ell(\lambda_{\mathsf{OWP}})$ is some polynomial in λ_{OWP}. We set $\lambda_{\mathsf{OWP}} = 2\mu$ and $\ell = \ell(\mu)$. We additionally require that this function is computable in NC^1. Such a function can be constructed assuming the subexponential time hardness discrete log assumption over $\mathbb{Z}_p^*$.

Sender Equivocal Oblivious Transfer: We require a sender equivocal oblivious transfer $\mathsf{OT} = (\mathsf{OT}_1, \mathsf{OT}_2, \mathsf{OT}_3)$ satisfying the properties in Definition 7. We will set $\lambda_{\mathsf{ot}} = 2\lambda$, and assume that the receiver security holds against adversaries of size polynomial in $2^{\lambda_{\mathsf{ot}}}$ and with maximum advantage of $2^{-\lambda_{\mathsf{ot}}}$. Such an OT can be built assuming subexponential time and advantage hardness of DDH.

Time Lock Puzzle: We require a time lock puzzle as in Definition 6. The TLP satisfies the following parameters.

- $\lambda_{\mathsf{TLP}} = 2\lambda$,
- $t_{\mathsf{TLP}} = \min(t, \sqrt{\mu})$. Looking ahead, for our MrNISC, we use $t = \lambda^{\Theta(1)(\log\log\lambda)^{-1}}$, in which case $t_{\mathsf{TLP}} = t$.
- The function $D(t_{\mathsf{TLP}}) = 2^{t_{\mathsf{TLP}}^{\epsilon}}$ for some constant $\epsilon > 0$.

Therefore, TLP with these parameters ensures the security against adversary of size polynomial in $2^{\lambda_{\mathsf{TLP}}}$ and depth bounded by $2^{t_{\mathsf{TLP}}^{\epsilon}}$ with the advantage bounded by $2^{-\lambda_{\mathsf{TLP}}}$. Further, Solve can be run by a circuit of depth $\mathsf{poly}(2^{t_{\mathsf{TLP}}}, \lambda_{\mathsf{TLP}})$.

Equivocal Garbled Circuits: We require a garbling scheme $\mathsf{Gb} = (\mathsf{Garble}, \mathsf{Eval}, \mathsf{GbEquiv})$ as described in Definition 8 for NC^1 satisfying the properties of correctness and equivocation. The security parameter will be set as ℓ defined above.

Theorem 5. *Assume that the following assumptions hold:*

- *A time lock puzzle as in Definition 6 exist,*
- DDH *over* $\mathbb{Z}_p^*$ *is subexponentially secure against adversaries of subexponential size,*

then, there exist a SEE *with the properties listed in Definition 23 as per parameters described in Definition 24.*

Construction. We describe the construction next. In the construction, we omit the security parameters. We also exhibit how by building a bit commitment. To commit to longer messages, $\mathsf{Com}_{2,C}$ described below is repeated in parallel.

$\mathsf{Com}_{1,R}(\mathbf{b}_R \in \{0,1\}^{\ell})$: Parse $\mathbf{b}_R = (b_1, \ldots, b_{\ell})$. Compute the following:

- Compute $\mathsf{ot}_{1,i} \leftarrow \mathsf{OT}_1(b_i; r_i)$ for $i \in [\ell]$ using independent randomness r_i,

- Compute $Z \leftarrow \mathsf{TLP.PGen}(\mathbf{b}_R, \mathbf{r})$, where $\mathbf{r} = (r_1, \ldots, r_\ell)$ used for generating ot_1 messages above,
- Output $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$.

$\mathsf{Com}_{1,C}(\mathbf{b}_C \in \{0,1\}^\ell)$: Compute and output $\mathsf{com}_{1,C} = \mathsf{OWP}(\mathbf{b}_C)$.

$\mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, m \in \{0,1\}; r', \{r'_i\}_{i\in[\ell]})$: Parse $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$. Let $H = H[\mathsf{com}_{1,C}, m] : \{0,1\}^\ell \to \{0,1\}$ be the circuit that takes as input $\mathbf{b} \in \{0,1\}^\ell$. It checks that $\mathsf{OWP}(\mathbf{b}) = \mathsf{com}_{1,C}$ and if so, it outputs m and 0 otherwise. Run the following steps.

- Run $\mathsf{Garble}(H; r') \to \Gamma, \mathsf{Lab}$,
- Compute $\mathsf{ot}_{2,i} = \mathsf{OT}_2(\mathsf{ot}_{1,i}, \mathsf{Lab}_{0,i}, \mathsf{Lab}_{1,i}; r'_i)$ for $i \in [\ell]$.
- Output $\mathsf{com}_{2,C} = \Gamma, \{\mathsf{ot}_{2,i}\}_{i\in[\ell]}$.

Remark 6. *The opening of* $\mathsf{com}_{2,C}$ *consist of* $(m, r', r'_1, \ldots, r'_\ell)$.

We now argue properties of the scheme.

Indistinguishability of $\mathsf{com}_{1,R}$: The indistinguishability property follows from the security of TLP and OT. We show this by indistinguishable hybrids. The first hybrid corresponds to the case when $\mathsf{com}_{1,R}$ is generated using $\mathbf{b}_R$, whereas the last hybrid corresponds to the case $\mathsf{com}_{1,R}$ is generated using 0^ℓ.
Hybrid$_0$: In this hybrid, we compute $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$ where: $\mathsf{ot}_{1,i} = \mathsf{OT}_1(b_i; r_i)$ for $i \in [\ell]$ and $Z = \mathsf{PGen}((\mathbf{b}_R, \mathbf{r}))$.

Hybrid$_1$: This hybrid is the same as the previous one except that we compute $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$ where: $\mathsf{ot}_{1,i} = \mathsf{OT}_1(b_i; r_i)$ for $i \in [\ell]$ and $Z = \mathsf{PGen}((0^\ell, \mathbf{r}'))$ where $\mathbf{r}'$ is independently sampled.

Claim 19. *For any adversary $\mathcal{A}$, of size polynomial in 2^λ and depth bounded by any polynomial $\mathsf{poly}(\lambda)$, it holds that:*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_0) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1]| \leq 2^{-(\lambda_{\mathsf{TLP}} = 2\lambda)}$$

This claim follows from the security of TLP. TLP is secure against adversaries of size polynomial in $2^{\lambda_{\mathsf{TLP}}}$, and depth $D(t_{\mathsf{TLP}}) \geq 2^{t^\epsilon_{\mathsf{TLP}}} \in \lambda^{\omega(1)}$. Thus one can form a reduction, distinguishing these two hybrids to breaking the security of TLP. Since $\lambda_{\mathsf{TLP}} = 2\lambda$, the claim holds.

Hybrid$_2$: This hybrid is the same as the previous one except that we compute $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$ where: $\mathsf{ot}_{1,i} = \mathsf{OT}_1(0; r_i)$ for $i \in [\ell]$ and $Z = \mathsf{PGen}((0^\ell, \mathbf{r}'))$ where $\mathbf{r}'$ is independently sampled.

Claim 20. *For any adversary $\mathcal{A}$, of size polynomial in $2^{\lambda_{\mathsf{ot}}}$, it holds that:*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_1) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1]| \leq \ell \cdot \ell \cdot 2^{-2\lambda}$$

This claim follows from the security of OT. OT is secure against adversaries of size polynomial in $2^{\lambda_{\mathsf{ot}}}$ with an advantage $2^{-\lambda_{\mathsf{ot}}}$. We make ℓ intermediate hybrids in which we switch one by one $\mathsf{ot}_{1,i}$ to be computed using 0 instead of b_i. Each intermediate hybrid is indistinguishable with an

advantage $2^{-\lambda_{\mathsf{ot}}}$. Since $\lambda_{\mathsf{ot}} = 2\lambda$, the claim holds.

Hybrid$_3$: This hybrid is the same as the previous one except that we compute $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$ where: $\mathsf{ot}_{1,i} = \mathsf{OT}_1(0; r_i)$ for $i \in [\ell]$ and $Z = \mathsf{PGen}((0^\ell, \mathbf{r}))$ where $\mathbf{r}$ is the randomness to compute $\{\mathsf{ot}_{1,i}\}_{i\in[\ell]}$.

Claim 21. *For any adversary $\mathcal{A}$, of size polynomial in $2^{2\lambda}$ and depth bounded by any polynomial $\mathsf{poly}(\lambda)$, it holds that:*

$$|\Pr[\mathcal{A}(\mathbf{Hybrid}_2) = 1] - \Pr[\mathcal{A}(\mathbf{Hybrid}_3) = 1]| \leq 2^{-2\lambda}$$

This claim follows from the security of TLP. TLP is secure against adversaries of size $2^{\lambda_{\mathsf{TLP}}}$, and depth $D(t_{\mathsf{TLP}}) \in \lambda^{\omega(1)}$. Thus one can form a reduction, distinguishing these two hybrids to breaking the security of TLP. Since $\lambda_{\mathsf{TLP}} = 2\lambda$, the claim holds.

Summing up, these three hybrids prove the required claim.

Verifiability of $\mathsf{com}_{1,C}$: This property is straightforward to observe. Observe that $\mathsf{Com}_{1,C}(\mathbf{b}) = \mathsf{OWP}(\mathbf{b})$. Since OWP has verifiable range of $\{0,1\}^\ell$, therefore $\mathsf{com}_{1,C}$ is verifiable.

Extraction when $\mathbf{b}_R = \mathbf{b}_C$: This property is also straightforward to observe and follows from the perfect correctness of OT, and the garbling scheme Gb. We define the Dec function. $\mathsf{Dec}(\mathbf{b}_R, \mathbf{r}, \mathsf{com}_{1,C}, \mathsf{com}_{1,R}, \mathsf{com}_{2,C})$: This algorithm parses $\mathbf{b}_R = (b_1, \ldots, b_\ell)$, $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$, $\mathbf{r} = (r_1, \ldots, r_\ell)$ and $\mathsf{com}_{2,C} = (\Gamma, \mathsf{ot}_{2,1}, \ldots, \mathsf{ot}_{2,\ell})$. It does the following:

- Run $\mathsf{Lab}'_{b_i,i} \leftarrow \mathsf{OT}_3(\mathsf{ot}_{2,i}, b_i, r_i)$ for $i \in [\ell]$,
- Output $\hat{m} \leftarrow \mathsf{Eval}(\Gamma, \{\mathsf{Lab}'_{\mathbf{b}_R}\})$.

The correctness is straightforward to observe. Parse $\mathbf{r}' = (r', r'_1, \ldots, r'_\ell)$. Let $\Gamma, \mathsf{Lab} = \mathsf{Garble}(H; r')$ where $H[\mathsf{Com}_{1,C}(\mathbf{b}_R), m]$ for some message m. Let $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$ where $\mathsf{ot}_{1,i} = \mathsf{OT}_1(b_i, r_i)$ and $\mathsf{ot}_{2,i} = \mathsf{OT}_2(\mathsf{ot}_{1,i}, \mathsf{Lab}_{0,i}, \mathsf{Lab}_{1,i}; r'_i)$ for $i \in [\ell]$. Our first observation is that $\mathsf{Lab}'_{b_i,i} = \mathsf{Lab}_{b_i,i}$ for all $i \in [\ell]$ due to perfect correctness of OT. Therefore $\hat{m} = \mathsf{Eval}(\Gamma, \{\mathsf{Lab}_{\mathbf{b}_R}\})$. Due to perfect correctness of garbled circuit we have that $\mathsf{Eval}(\Gamma, \{\mathsf{Lab}_{\mathbf{b}_R}\}) = H[\mathsf{Com}_{1,C}(\mathbf{b}_R), m](\mathbf{b}_R)$. This is equal to m, by definition of H.

Hard to force $\mathbf{b}_R = \mathbf{b}_C$ by adversaries in $\mathcal{C}_{\mathcal{A}}$. This follows from the reduction to the security of OWP, and the fact that Solve runs in time polynomial in $2^{t_{\mathsf{TLP}}}$. Let $\mathcal{A}$ be an adversary that wins in the security game for this property and is of the size polynomial in 2^μ with advantage more than $2^{-\mu}$. Then, we show how to build a reduction that runs in size polynomial in $2^{\lambda_{\mathsf{OWP}}}$ and wins in the breaking the security of OWP with the same advantage.

- The reduction receives as input $\mathsf{com}_{1,C} = \mathsf{OWP}(\mathbf{b})$ for a randomly chosen $\mathbf{b} \leftarrow \{0,1\}^\ell$.
- The reduction sends to the adversary $\mathcal{A}$ $\mathsf{com}_{1,C}$ and receives $\mathsf{com}_{1,R}$ formatted as $\mathsf{ot}_{1,1}(b'_1, r'_1), \ldots, \mathsf{ot}_{1,\ell}(b'_\ell, r'_\ell), Z$ $\mathsf{PGen}(\mathbf{b}', r')$.
- The reduction solves Z using a circuit size polynomial in $\mathsf{poly}(2^{t_{\mathsf{TLP}}}) \leq 2^{\frac{\mu}{2}}$ and recovers $\mathbf{b}', \mathbf{r}'$.
- It outputs $\mathbf{b}'$ if $\mathsf{com}_{1,C} = \mathsf{OWP}(\mathbf{b}')$.

Note that the view of $\mathcal{A}$ is identical to the view in the required security property of Com. If $\mathcal{A}$ produces $\mathsf{com}_{1,R}$ using $\mathbf{b}'$ that equals to the random challenge $\mathbf{b}$, then the reduction successfully recovers it by breaking TLP in time $2^{\mu/2}$. If the size of the adversary $\mathcal{A}$ is polynomial in 2^μ, the size of the reduction is also polynomial in $2 \cdot 2^\mu$ which is a contradiction as $\lambda_{\mathsf{OWP}} = 2\mu$.

Equivocation with $\mathbf{b}_R \neq \mathbf{b}_C$. We describe our algorithm $\mathcal{S}$ and then prove that it runs in time polynomial $2^{t_{\mathsf{TLP}}}$ and satisfies the equivocation property.

$\mathcal{S}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathbf{r}', m)$: Parse $\mathsf{com}_{1,R} = (\mathsf{ot}_{1,1}, \ldots, \mathsf{ot}_{1,\ell}, Z)$, $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_C)$ and $\mathsf{com}_{2,C} = \Gamma, \{\mathsf{ot}_{2,i}\}_{i\in[\ell]}$. Recall, how are each of the strings generated in the equivocation game. $\mathsf{com}_{1,R}$ is generated by computing: For $i \in [\ell]$, $\mathsf{ot}_{1,i} = \mathsf{OT}_1(\mathbf{b}_{R,i}; r_i)$ using some randomness r_i and Z is generated by computing $\mathsf{PGen}(\mathbf{b}_R, \mathbf{r} = (r_1, \ldots, r_\ell))$. Receiver's randomness may be arbitrarily chosen. For the committer, $\mathsf{com}_{2,C}$ is generated honestly by committing to 0 using honestly generated randomness $\mathbf{r}'$. Parse $\mathbf{r}' = (r', r'_1, \ldots, r'_\ell)$. Γ, Lab is computed as $\mathsf{Garble}(H[\mathsf{com}_{1,C}, 0]; r')$. Then we compute $\mathsf{ot}_{2,i} = \mathsf{OT}_2(\mathsf{ot}_{1,i}, \mathsf{Lab}_{0,i}, \mathsf{Lab}_{1,i}; r'_i)$ for $i \in [\ell]$. Finally $\mathsf{com}_{2,C} = (\Gamma, \{\mathsf{ot}_{2,i}\}_{i\in[\ell]})$. Thus to equivocate, compute the following steps:

- Run $\mathsf{Solve}(Z) = (\mathbf{b}_R, \mathbf{r})$.
- Equivocate Garbled Circuit: Run $\mathsf{GbEquiv}(\Gamma, \mathsf{Lab}_{\mathbf{b}_R}, H[\mathsf{com}_{1,C}, m], \mathbf{b}_R) \to \mathsf{Lab}', s$ where Lab' is the new set of labels and s is the randomness that explains $\mathsf{Garble}(H[\mathsf{com}_{1,C}, m]; s) \to \Gamma, \mathsf{Lab}'$. Further $\mathsf{Lab}'_{\mathbf{b}_R} = \mathsf{Lab}_{\mathbf{b}_R}$.
- Equivocate ot_2: For $i \in [\ell]$, compute $s_i = \mathsf{OT.Equiv}(\mathbf{b}_{R,i}, r_i, \mathsf{ot}_{2,i}, r'_i, \mathsf{Lab}'_{0,i}, \mathsf{Lab}'_{1,i})$.
- Output $(m, \mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathsf{com}_{2,C}, \mathbf{s} = (s, s_1, \ldots, s_\ell))$.

The run time of the simulator above is polynomial in $2^{t_{\mathsf{TLP}}}$ which is polynomial in 2^t as per the setting of the parameters. The proof of security is immediate and follows from the equivocability property of the garbled circuit and OT. We show this by identical hybrids. The first hybrid corresponds to the case when m is committed, and the last hybrid corresponds to the simulator, where 0 is committed first and then equivocated to m.

Hybrid$_0$: In this hybrid, compute $\mathsf{com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, m; \mathbf{r}')$. Output $(m, \mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathsf{com}_{2,C}, \mathbf{r}')$.

Hybrid$_1$: In this hybrid, we use the equivocation of the garbled circuit property. First generate $\Gamma, \mathsf{Lab} \leftarrow \mathsf{Garble}(H[\mathsf{com}_{1,C}, 0]; r')$. Observe that $H[\mathsf{com}_{1,C}, 0](\mathbf{b}_R) = H[\mathsf{com}_{1,C}, m](\mathbf{b}_R) = 0$. Therefore, due to the equivocation property of the garbled circuits, we can compute $\mathsf{GbEquiv}(\Gamma, \mathsf{Lab}_{\mathbf{b}_R}, H[\mathsf{com}_{1,C}, m], \mathbf{b}_R) \to \mathsf{Lab}', s$. We set $\mathsf{com}_{2,C} = \Gamma$ and $\{\mathsf{ot}_{2,i} = \mathsf{OT}_2(\mathsf{ot}_{1,i}, \mathsf{Lab}'_{0,i}, \mathsf{Lab}'_{1,i}; r'_i)\}_{i\in[\ell]}$. Output $(m, \mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathsf{com}_{2,C}, (s, r'_1, \ldots, r'_\ell))$.

The two distributions above are identical due to the equivocation property of the garbled circuits.

Hybrid$_2$: In this hybrid, we use the equivocation property of OT. We first generate $\mathsf{ot}_{2,i} = \mathsf{OT}_2(\mathsf{ot}_{1,i}, \mathsf{Lab}_{0,i}, \mathsf{Lab}_{1,i}; r'_i)$ for $i \in [\ell]$. Then, since $\mathsf{com}_{1,R}$ consists of OT_1 messages corresponding to $\mathbf{b}_R \neq \mathbf{b}_C$, we can equivocate $\mathsf{ot}_{2,i}$ as follows. We run $s_i = \mathsf{OT.Equiv}(\mathbf{b}_{R,i}, r_i, \mathsf{ot}_{2,i}, r'_i, \mathsf{Lab}'_{0,i}, \mathsf{Lab}'_{1,i})$. This can be done because $\mathsf{Lab}'_{b_{R,i},i} = \mathsf{Lab}_{b_{R,i},i}$. Thus at the end of this we have randomness s_i such that $\mathsf{ot}_{2,i} = \mathsf{OT}_2(\mathsf{ot}_{1,i}, \mathsf{Lab}'_{0,i}, \mathsf{Lab}'_{1,i}; s_i) = \mathsf{OT}_2(\mathsf{ot}_{1,i}, \mathsf{Lab}_{0,i}, \mathsf{Lab}_{1,i}; r'_i)$. Output of this hybrid is $(m, \mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathsf{com}_{2,C}, \mathbf{s})$ where $\mathbf{s} = (s, s_1, \ldots, s_\ell)$.

This hybrid is identical to the previous hybrid due to the security of OT.

9 Construction of Reusable Statistical ZK arguments with Sometimes Statistical Soundness

In this section, we construct our zk protocol. Before we do that we give an overview of this construction.

9.1 Overview

In this section, we give a brief overview of the zk construction. Recall that we want to construct a two round (delayed instance) zk with SPS simulation, while still being perfectly sound with some probability. Further, the first round should be reusable across sessions.

Our starting point is the SPS ZK protocol/statistical ZAP arguments of [BFJ+20, GJJM20]. The protocol rely's on the following primitives:

- A correlation intractable hash function $\mathcal{H}(K, \star) \to \{0,1\}^{\ell}$ [CCH+19, PS19],
- A two-round statistically hiding sometimes extractable commitment $\mathsf{Com} = (\mathsf{Com}_{1,R}, \mathsf{Com}_{2,C})$ [KK19],

A (somewhere) statistically correlation intractable function is associated with an algorithm $\mathsf{FakeGen}$ that takes as input a polynomial time computable function $f : \{0,1\}^{\ell_{in}} \to \{0,1\}^{\ell}$, and outputs a key K_f, for which there does not exist in input $x \in \{0,1\}^{\ell_{in}}$ such that $\mathcal{H}(K_f, x) = f(x)$. These functions can be built from LWE. Further, $\mathsf{FakeGen}$ produces pseudorandom outputs, and thus key K_f hides f computationally.

A two-round statistically hiding sometimes extractable commitment scheme on the other hand, has the following structure.

- In the first round, the receiver samples a $\mathbf{b}_R \in \{0,1\}^{\mu}$ and computes and outputs $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_R; r_R)$.
- In the second round, the committer samples $\mathbf{b}_C \in \{0,1\}^{\mu}$ randomly, and outputs any number of commitments $\mathbf{b}_C, \{\mathsf{com}_{2,C,i} = \mathsf{Com}_{2,C}(\mathbf{b}_C, \mathsf{com}_{1,R}, m_i)\}_{i \in [T]}$.

The protocol has the following property. If $\mathbf{b}_R \neq \mathbf{b}_C$ (or if $\mathsf{com}_{1,R}$ is not well formed as per the protocol), then, the honestly generated commitments $\mathsf{com}_{2,C}$, statistically hide the messages $\{m_i\}_{i \in [T]}$. On the other hand, if, $\mathbf{b}_R = \mathbf{b}_C$, then there exists an efficient algorithm Dec such that: $\mathsf{Dec}(\mathbf{b}_R, r_R, \mathsf{com}_{2,C,i}) = m_i$ for $i \in [T]$. Further, an honest receiver can ensure that $\mathbf{b}_C = \mathbf{b}_R$ with probability at least $\Omega(2^{-\mu})$, and to an adversarial polynomial time committer the view is indistinguishable from the view when $\mathbf{b}_C \neq \mathbf{b}_R$. The works of [KKS18] showed that such commitments can be built from assumptions such as LWE or DDH.

Once we have these primitives, then the SPS ZK protocol of [BFJ+20], follows the following template. Let (x, w) be the isntance witness pair.

- In the first round, the verifier chooses $\mathbf{b}_V \leftarrow \{0,1\}^{\mu}$, and outputs $\mathsf{zk}_{1,P} = (\mathsf{com}_{1,R}, K)$ where $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_V)$ and $K \leftarrow \mathcal{H}.\mathsf{FakeGen}(f)$ for some function f described later,
- In the second round, the prover samples $\mathbf{b}_P \leftarrow \{0,1\}^{\mu}$, and then computes $\mathsf{com}_{2,C,i} = \mathsf{Com}_{2,C}(\mathbf{b}_P, \mathsf{com}_{1,R}, a_i; r'_i)$ for $i \in [N]$ where $(a_1, \ldots, a_N)$ are the values committed to during a special Σ protocol[4] for proving x. Then,
  - The prover runs $\mathcal{H}(K, (x, \mathbf{b}_P, \mathsf{com}_{2,C})) = \mathbf{e}$,
  - Outputs commitments $\mathsf{com}_{2,C} = \{\mathsf{com}_{2,C,i}\}_{i \in [N]}$ along with openings $\mathsf{com}_2, \{a_i, r'_i\}_{i \in \mathsf{Set}}$ where Set is the set dictated by the challenge $\mathbf{e}$ of the Σ protocol.

[4] As an example, think of it as the Blum's Hamiltonicity Protocol. As in the construction of NIZK from LWE[CCH+19, PS19], it suffices to use a parallel repetion of a sigma protocol for NP with i) 1/2-special soundness, ii) efficient $\mathsf{BadChallenge}$ computation.

The statistical WI property follows from the fact that when $\mathbf{b}_P \neq \mathbf{b}_C$, then $\mathsf{com}_{2,C}$ are statistically hiding. One needs more work to prove that it is actually SPS ZK by using an inefficient equivocator of $\mathsf{com}_{2,C}$ with a simulator of the Σ protocol. For the soundness property, observe that when $\mathbf{b}_V = \mathbf{b}_P$, then the commitments are binding to the value computed by $\mathsf{Dec}(\mathbf{b}_V, r_R, \star)$. We exploit this to set f as follows. We set f to be the function that computes $\mathbf{e}^* = \mathsf{BadChallenge}(x, a_1, \ldots, a_N)$ where $(a_1, \ldots, a_N)$ is recovered by running $\mathsf{Dec}(\mathbf{b}_V, r_R, \star)$.

Perfect Soundness Mode. The protocol above does not have a perfect soundness mode. However it turns out that in the simultaneous message model, there is a very simple modification of the protocol above that gives us a perfect soundness mode. The modification is described as follows.

- In the first round, the verifier outputs $\mathsf{zk}_{1,V} = (\mathsf{com}_{1,R}, K)$ as before, but the prover outputs $\mathsf{zk}_{1,P} = \mathbf{b}_P$ in the clear.
- In the second round, the prover outputs as before, but using $\mathbf{b}_P$ displayed in the first round itself.

The reason why this protocol has a perfect soundness mode is because, $\mathbf{b}_P$ is displayed in the first round itself, and so the the first round already determines the if the prover can cheat in the second round or not. Unfortunately, the problem with this naive approach is that it fails in our setting where the same prover/verifier first message can be used over and over with multiple verifiers/receivers. In fact, it even fails when an honest prover interacts with a rushing malicious verifier. If such a verifier sees $\mathbf{b}_P$, then it can choose $\mathbf{b}_V = \mathbf{b}_P$, which will put the prover in the perfect soundness mode, and its proofs will no longer be simulatable.

Our First Idea: A different criteria for soundness Imagine, if we could modify the criteria for the soundness mode as follows. In this model, $\mathsf{zk}_{1,P}$ is $\alpha = \mathsf{OWP}(\mathbf{b}_P)$ for a one-way permutation as opposed to $\mathbf{b}_P$ in the clear, and $\mathsf{zk}_{1,V}$ is as before, $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_V)$. As before the perfect soundness must hold if $\mathbf{b}_V = \mathbf{b}_P$ and perfect zero-knowledge otherwise. This high-level approach appears to make sense, as intuitively a verifier must compute $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathsf{OWP}^{-1}(\alpha))$ to violate soundness.

To work this idea in the reusable setting, we have to tackle one-more issue. We need to make sure that $\mathsf{zk}_{2,P}$ must not reveal information about $\mathbf{b}_P$ as in the reusable setting, one can choose $\mathsf{zk}'_{1,V}$ after seeing a second message $\mathsf{zk}_{2,P}$ used in some other session. We make this intuition formal by this abstraction called "Sometimes Extractable Equivocal Commitments" or SEE.

Sometimes Extractable Equivocal Commitments. A SEE scheme consists of three algrorithms $(\mathsf{Com}_{1,R}, \mathsf{Com}_{1,C}, \mathsf{Com}_{2,C})$ and is a commitment scheme that captures the issues pointed above in the simultaneous message model. In the first round,

- The receiver chooses $\mathbf{b}_R$ and computes and outputs $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_R; r)$,
- The committer chooses $\mathbf{b}_C$ and computes and outputs $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_R)$ deterministically. Further, $\mathsf{com}_{1,C}$ is essentially a one-way permutation.

In the second round, the committer outputs $\mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, m; r')$ which does not depend on the private state of the committer. We want similar properties as before: If $\mathbf{b}_R = \mathbf{b}_C$, then the commitment is fully extractable, wheras when $\mathbf{b}_R \neq \mathbf{b}_C$, then $\mathsf{com}_{2,C}$ is statistically hiding (and infact equivocable). Other than that, we will ensure that it is computationally hard for an adversarial receiver to create $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_C)$ for $\mathbf{b}_C$ chosen by the committer even, after

seeing $\mathsf{com}_{1,C}$. Plugging in this commitment scheme with a correlation intractable hash function gives rise to the following zk protocol.

- In the first round, the verifier outputs $\mathsf{zk}_{1,V} = (\mathsf{Com}_{1,R}(\mathbf{b}_V; r), K)$ as before and the prover outputs $\mathsf{zk}_{1,P} = \mathsf{Com}_{1,C}(\mathbf{b}_P)$.
- In the second round, the prover computes $\mathsf{com}_{2,C,i} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, a_i; r'_i)$ for $i \in [N]$ where $(a_1, \ldots, a_N)$ are the values committed to during a special Σ protocol. Then,
  - The prover runs $\mathcal{H}(K, (x, \mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathsf{com}_{2,C})) = \mathbf{e}$,
  - Outputs commitments $\mathsf{com}_{2,C} = \{\mathsf{com}_{2,C,i}\}_{i\in[N]}$ along with openings $\mathsf{com}_2, \{a_i, r'_i\}_{i\in\mathsf{Set}}$ where Set is the set dictated by the challenge $\mathbf{e}$ of the Σ protocol.

Observe that now, the protocol has a perfect soundness mode, namely when $\mathbf{b}_P = \mathbf{b}_V$. Further, the verifier message is reusable across multiple prover sessions as the soundness holds with same probability across multiple sessions. On the other hand, the prover's first message $\mathsf{zk}_{1,V} = \mathsf{Com}_{1,C}(\mathbf{b}_V)$ is also reusable with different verifiers, as it is computationally hard to produce $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_V)$ with the $\mathbf{b}_P = \mathbf{b}_V$.

Issue with Complexity Hierarchy wrt MrNISC. Although if we have a committment scheme like this the approach almost works, we must identify how this fits in the bigger scheme of things with other primitives in the MrNISC scheme. Our main observation is the following. In the protocol, we are also using a receiver assisted one-round CCA commitment (aCCA), and that protocol is intimately tied with the zk we are trying to build. As pointed out in Section 2.2.4, on one hand we need the zk to be sound against circuits that can perform CCAVal, on the other hand, aCCA commitments need to be secure against circuits that are capable of running zk SPS simulator. This might feel like a deadlock, nevertheless, we introduce a new-axis of hardness.

We will use commitments aCCA which are secure against circuits of some quasipolynomial size such that $\mathsf{aCCA.CCAVal}$ runs in polynomial depth but size 2^{λ^c} for $c > 0$. In the zk we build:

- Soundness holds against adversaries of $\mathsf{poly}(\lambda)$ depth and size $2^{\lambda^{c_2}}$.
- The zk simulator can be implemented by a circuit of size/and depth $T_{\mathsf{zk},\mathcal{S}}$ against which CCA security holds.

We do this by incorporating time-lock puzzle like properties in our commitment scheme and hence the zk protocol.

Summing up. Summing up, as a first step we build an SEE scheme described above with a few additional properties. Below we list all the properties. The only new addition to what was described before is that $\mathsf{com}_{2,C}$ can be equivocated in polynomial time given the opening $\mathbf{b}_R, r$ of $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_R; r)$.

- Extractability: If $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_R; r)$ and $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_R)$, then $\mathsf{com}_{2,C} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, m)$ is polynomial time extractable using Dec algorithm. This property is identical to the one described before.
- Equivocability: If $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_R; r)$ and $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_C)$ where $\mathbf{b}_C \neq \mathbf{b}_R$, then there exists a polynomial time algorithm $\mathsf{SEE}.\mathcal{S}$ that takes as input $\mathbf{b}_R, r, \mathsf{com}_{2,C}, m, r'$ where $\mathsf{com}_{2,C} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, 0, r')$ and outputs an opening s' such that $\mathsf{com}_{2,C} =$

$\mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, m; s')$. Further, $\mathsf{com}_{2,C}, s', m$ generated this way is identical to the case when $\mathsf{com}_{2,C}$ was a commitment of m and s' was its opening. This is stronger than statistical indistinguishability.

- Indistinguishability of $\mathbf{b}_R$: We require that an $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_R)$ hides $\mathbf{b}_R$. Further, for any computationally bounded committer $\mathbf{b}_C = \mathbf{b}_R$ with a probability of $2^{-\Omega(\mu)}$. We also require that the distribution of transcripts when this event happens are indistinguishable to when this event does not happen.

- Hard to force $\mathbf{b}_R = \mathbf{b}_C$: We require that a computationally bounded adversarial receiver given $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_C)$ for a randomly chosen $\mathbf{b}_C$ cannot come up with $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_C)$ with all but negligible probability.

We build such an SEE scheme relying on DDH assumption over $\mathbb{Z}_p^*$ in Section 8.4.

Once we have such a commitment scheme, we can solve all problems, except we need to control the size of the circuit that runs $\mathsf{zk}.\mathcal{S}$ by a quasi-polynomial sized circuit. Our main idea to get around this is to use a time-lock puzzle. We add $Z = \mathsf{TLP}(\mathbf{b}_V, r)$ to $\mathsf{zk}_{1,V}$. The TLP parameters are set so that it is broken by a quasipolynomial sized circuit, but it is secure against all circuits of poly depth of size 2^{λ^c}.

Therefore in our modified protocol:

- In the first round, the verifier outputs $\mathsf{zk}_{1,V} = (Z = \mathsf{TLP}(\mathbf{b}_V, r), \mathsf{Com}_{1,R}(\mathbf{b}_V; r), K)$ and the prover outputs $\mathsf{zk}_{1,P} = \mathsf{Com}_{1,C}(\mathbf{b}_P)$.

- In the second round, the prover computes $\mathsf{com}_{2,C,i} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, a_i; r_i')$ for $i \in [N]$ where $(a_1, \ldots, a_N)$ are the values committed to during a special Σ protocol. Then,
  - The prover runs $\mathcal{H}(K, (x, \mathsf{com}_{1,R}, \mathsf{com}_{1,C}, \mathsf{com}_{2,C})) = \mathbf{e}$,
  - Outputs commitments $\mathsf{com}_{2,C} = \{\mathsf{com}_{2,C,i}\}_{i\in[N]}$ along with openings $\mathsf{com}_2, \{a_i, r_i'\}_{i\in\mathsf{Set}}$ where Set is the set dictated by the challenge $\mathbf{e}$ of the Σ protocol.

This is really useful, and in particular, we can now simulate zk by first breaking Z to learn $\mathbf{b}_V, r$ and then using the equivocator of the commitments and the simulator of the Σ protocol to simulate the second message.

In our construction, to make the construction mode modular, we incorporate the TLP aspect in the SEE scheme (see Section 8.4) and not in our zk protocol. In our commitment scheme, the equivocation property is required to hold only against a receiver which generates $\mathsf{com}_{1,R}$ using the honest algorithm (although with adversarial randomness). This brings us to our last issue.

One Last Issue. This solves all the issues, except that the simulator fails completely if a verifier does not generate $\mathsf{com}_{1,R}$ as per the specification of the protocol. Indeed, Z may not be a time lock puzzle and give the randomness needed by the simulator to equivocate $\mathsf{com}_{2,C}$. To fix this issue, the verifier now supplies a simultaneous message non-interactive distributional indistinguishability proof NIDI (see Section 8.1 for details about NIDI) proving that the verifier messages are well formed as in the protocol described above. This soundness property of this proof system guarantees that the verifier messages are well-formed, which is useful for the simulator, and the distibutional indistinguishability guarantees that $\mathsf{zk}_{1,V}$ generated using $\mathbf{b}_V$ is computationally indistinguishable to $\mathsf{zk}_{1,V}$ generated using 0^μ. The analysis of the protocol and setting up parameters requires some care, and we describe it formally next.

9.2 Construction

In this section, we construct a reusable statistical ZK arguments with sometimes statistical soundness (henceforth denoted by $\mathsf{zk} = (\mathsf{ZKProve}_1, \mathsf{ZKVerify}_1, \mathsf{ZKProve}_2, \mathsf{ZKVerify}_2)$) as defined in Section 5.1. We now give the parameters associated the various adversary classes that we will guarantee security for. We will then follow it up with the parameters for the underlying primitives we use. Let λ be the security parameter for zk.

Definition 25 (Parameters of zk)**.** *We achieve* zk *for the following parameters.*

- *For the soundness property the parameters* $(\mathcal{C}_{sound}, \epsilon_{sound,1}, \epsilon_{sound,2})$ *we achieve will be as follows.* $\mathcal{C}_{sound}$ *consists of circuits with any* $\mathsf{poly}(\lambda)$ *depth Boolean circuits of size bounded by any polynomial in* 2^λ*. We will set* $\epsilon_{sound,1} = 2^{-\ell} = \Omega(2^{-\ell_\mu})$ *for some polynomial* $\ell(\lambda)$ *and* $\epsilon_{sound,2} = 2^{-\lambda}$*.* ℓ_μ *is defined when we define the parameters for* SEE *scheme.*
- *For the zero knowledge,* $\mathcal{C}_{\mathcal{S}}, \mathcal{C}_{\mathsf{zk}}, \epsilon_{\mathcal{S}}$ *are set as follows.* $\mathcal{C}_{\mathcal{S}}$ *is the complexity class of the simulator.* $\mathcal{C}_{\mathcal{S}}$ *consists of circuits of size* 2^{λ^ρ} *for some parameter* $\rho \in \Theta(1) \log\log \lambda^{-1}$*, which can be chosen as a parameter to the scheme. We will also set* $\mathcal{C}_{\mathsf{zk}}$*, which is the class of the zero-knowledge verifier to be the same as* $\mathcal{C}_{sound}$ *of* $\mathsf{poly}(\lambda)$ *depth circuits of size polynomial in* 2^λ*.* $\epsilon_{\mathcal{S}}$ *will be set as* $2^{-\lambda}$*.*

Used Primitives. We make use of the following primitives and instantiated with the following parameters. These instantiated parameters for the primitives we use are loose for what we require.

NIDI Arguments: We require a NIDI scheme (Definition 18) as per the following specifications. Such a NIDI uses two security parameters $\lambda_{\mathsf{NIDI},S}$ and $\lambda_{\mathsf{NIDI},D}$. We set $\lambda_{\mathsf{NIDI},S} = \lambda$. We set $\mathcal{C}_{\mathsf{NIDI},S}$ to consist of all adversaries of size polynomial in $2^{\lambda_{\mathsf{NIDI},S}}$. We set $\epsilon_{\mathsf{NIDI},S} = 2^{-\lambda_{\mathsf{NIDI},S}}$. For this choice of $\lambda_{\mathsf{NIDI},S} = \lambda$, let $\ell_{\mathsf{NIDI}}(\lambda)$ be the length of τ's used in the scheme. We set $\lambda_{\mathsf{NIDI},D}$ as a polynomial in λ. This polynomial will ensure that the distributions we use, on input $\lambda_{\mathsf{NIDI},D}$ satisfy the following parameters:

- $\mathcal{C}_{\mathsf{NIDI},D}$ consists of all circuits of depth $\mathsf{poly}(\lambda)$ and size polynomial in 2^λ.
- $\epsilon_{\mathsf{NIDI},D} = 2^{-\ell_{\mathsf{NIDI}} \cdot \ell_\mu \lambda}$. ($\ell_\mu$ is defined along with the instantiation for the sometimes extractable equivocal scheme).

Further, this setting will ensure that:

- $\mathcal{C}_{\mathsf{NIDI},DI} = \mathcal{C}_{\mathsf{NIDI},D}$.
- $\epsilon_{\mathsf{NIDI},DI} = O(\epsilon_{\mathsf{NIDI},D} 2^{\ell_{\mathsf{NIDI}}})$.

As shown in Theorem 4, this can be constructed assuming subexponential security of iO, a time lock puzzle scheme (Definition 6), and subexponential time and advantage security of the SXDH assumption.

Sometimes Extractable Equivocal Commitments: We use three parameters λ_{com}, μ_{com}, and t_{com}, as follows.

- We set $\mu_{\mathsf{com}} = \lambda$. This ensures that $\mathcal{C}_{\mathcal{A},\mathsf{com}}$ consists of all circuits of size polynomial in 2^λ. Let $\ell_\mu(\lambda)$ be the length of the challenges $\mathbf{b}_R$ to support this.
- We set $t_{\mathsf{com}} = \lambda^\rho$. This ensures the commitments are extractable in size polynomial in $2^{t_{\mathsf{com}}}$.

- We set $\lambda_{\mathsf{com}} = \ell_{\mathsf{NIDI}}(\lambda)\ell_\mu(\lambda)\lambda$. This choice ensures that $\mathcal{C}_{\mathsf{com},D}$ consists of all circuits of size polynomial in $2^{\lambda_{\mathsf{com}}}$ and depth polynomial in λ_{com}. This ensures $\epsilon_{\mathsf{com},D} = 2^{-\lambda_{\mathsf{com}}}$.

As shown in Theorem 5, can be constructed assuming subexponential security of iO, a time lock puzzle scheme (Definition 6), and subexponential time and advantage security of the DDH over $\mathbb{Z}_p^*$.

Σ-PROTOCOL: We use a statistically sound Σ protocol for NP, which is a parallel repetition of the following basic protocol. Assume that the length of the instance is a fixed polynomial in λ. We will build our zk protocol for the same length instances.

- The first message $\Sigma_1(x, w)$ by the prover consists of non-interactive commitments of some messages $a_1, \ldots, a_N \in \{0,1\}^{N(\lambda)}$. We define $\Sigma.\mathsf{SampFirst}$ to mean the algorithm that outputs $a_1, \ldots, a_N$.
- In the second round, the verifier outputs a bit $e \in \{0,1\}$.
- In the third round, the prover outputs z which consists of opening of some subset of the commitments based on the challenge bit e. Verifier accepts or rejects based on the transcript.

The protocol satisfies a number of different properties. The first property is related to the soundness and the second property to the zero-Knowledge property of the protocol.

- When x is unsatisfiable, then, given any $a_1 \ldots, a_N$ an accepting proof of at most one out of two choice of $e \in \{0,1\}$ can exist. We call this as the $\mathsf{BadChallenge}$. We assume that computing $\mathsf{BadChallenge}$ can be done by an NC^1 function Bad that takes x and $a_1, \ldots, a_N$ as the input.
- The protocol satisfies honest-verifier zero knowledge property. That is, given $e \in \{0,1\}$, for any x, one can efficiently sample $\Sigma.\mathcal{S}(e, x) \to z' = \mathsf{Set}, \{a'_i\}_{i\in\mathsf{Set}}$. The protocol ensures that the distribution of $\{a'_i\}_{i\in\mathsf{Set}}, e$ is identical to the case when $a_1, \ldots, a_N$ were committed to using an honest proof and then the prover gives out $(z = \mathsf{Set}, \{a_i\}_{i\in\mathsf{Set}}, e)$.

Looking ahead, we will compile such a protocol to a zk. The commitment we will use will be a sometimes extractable equivocal commitments.

CORRELATION INTRACTABILITY HASH FUNCTION: We require a CI hash function $\mathcal{H} = (\mathsf{FakeGen}, \mathsf{Eval})$ (see Definition 5). We set $\lambda_{ci} = \ell_{\mathsf{NIDI}} \cdot \ell_\mu\lambda$. This ensures that the hash keys corresponding to two functions are distinguishable to circuits of size polynomial in $\mathcal{C}_{ci} = 2^{\lambda_{ci}}$ with advantage at most $\epsilon_{ci} = 2^{-\lambda_{ci}}$. Finally, for this choice of parameters, there exists a polynomial $\ell_{ci}(\lambda_{ci})$ such that the security holds for functions of bounded depth (say λ_{ci}) with $\ell_{ci}(\lambda_{ci})$ output bits. We use this as the parallel repetition parameter for the Σ protocol.

This can be constructed assuming subexponential time and advantage hardness of LWE [PS19].

DISTRIBUTION $\mathcal{D}_{\mathbf{b}_R}$: For $\mathbf{b}_R \in \{0,1\}^{\ell_\mu}$, we define the distribution $\mathcal{D}_{\mathbf{b}_R}$ as follows.

- Sample $\mathsf{com}_{1,R} \leftarrow \mathsf{Com}_{1,R}(\mathbf{b}_R; \mathbf{r})$.
- Sample $K \leftarrow \mathcal{H}.\mathsf{FakeGen}(f[\mathbf{b}_R, \mathbf{r}])$, where $f : \{0,1\}^* \to \{0,1\}^{\ell_{ci}}$ is a function described below.

Observe that for any two $\mathbf{b}_1$ and $\mathbf{b}_2$ in $\{0,1\}^{\ell_\mu}$, $\mathcal{D}_{\mathbf{b}_1}$ and $\mathcal{D}_{\mathbf{b}_2}$ are $O(\ell_\mu \cdot (2^{-\lambda_{ci}} + 2^{-\lambda_{\mathsf{com}}})) = O(2^{-\ell_\mu \cdot \ell_{\mathsf{NIDI}} \cdot \lambda})$ indistinguishable to circuits of depth polynomial in λ but size $2^{\lambda_{\mathsf{com}}}$. Let L_{NIDI} denote the language supporting these distributions $\mathcal{D}_{\mathbf{b}}$ for all $\mathbf{b}$.

FUNCTION $f : \{0,1\}^* \to \{0,1\}^{\ell_{ci}}$: takes as input $(x, \mathsf{com}_{1,C}, \mathsf{com}_{1,R}, \mathsf{com}_{2,C} = (\mathsf{com}_{2,C,1}, \ldots, \mathsf{com}_{2,C,N\cdot\ell_{ci}}))$.

- It partitions $\mathsf{com}_{2,C}$ into ℓ_{ci} chunks. Each chunk is $(\mathsf{com}_{2,C,j\cdot N+1}, \ldots, \mathsf{com}_{2,C,(j+1)N})$ for $j \in [0, \ell_{ci} - 1]$.
- It decrypts each chunk using $\mathsf{Com.Dec}$ using its private state $\mathbf{b}_R, \mathbf{r}$. Let us say that each chunk decrypts to $a_{jN+1}, \ldots a_{(j+1)N}$.
- Then it computes $\mathsf{Bad}(x, a_{jN+1}, \ldots a_{(j+1)N}) = e_{j+1}$.
- Finally it outputs $\mathbf{e} = (e_1, \ldots, e_{\ell_{ci}})$.

Thus, we have the following theorem:

Theorem 6. *Assume that the following assumptions hold:*

- *A subexponentially secure indistinguishability obfuscator exists,*
- *A time lock puzzle as in Definition 6 exist,*
- *SXDH is subexponentially secure against adversaries of subexponential size,*
- *DDH over $\mathbb{Z}_p^*$ is subexponentially secure against adversaries of subexponential size,*
- *Subexponential time and advantage security of* LWE.

then, there exist a reusable statistical ZK argument with sometimes statistical soundness as Defined in Section 13. For this scheme, the complexity parameters are defined in Definition 25.

We now describe our construction.

Construction.

$\underline{\mathsf{ZKProve}_1(1^\lambda)}$: Compute the following steps.

- Sample $\tau \leftarrow \{0,1\}^{\ell_{\mathsf{NIDI}}}$ and $\mathbf{b}_P \leftarrow \{0,1\}^{\ell_\mu}$.
- Compute $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_P)$.
- Output $\mathsf{zk}_{1,P} = (\tau, \mathsf{com}_{1,C})$.

$\underline{\mathsf{ZKVerify}_1(1^\lambda)}$: Compute the following steps.

- Sample $\mathbf{b}_V \leftarrow \{0,1\}^{\ell_\mu}$.
- Compute $\Pi \leftarrow \mathsf{NIDI.P}(\mathcal{D}_{\mathbf{b}_V})$.
- Output $\mathsf{zk}_{1,V} = \Pi$.

$\underline{\mathsf{ZKProve}_2(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,P}, x, w)}$: Compute the following steps.

- Parse $\mathsf{zk}_{1,V} = \Pi$ and $\mathsf{zk}_{1,P} = (\tau, \mathsf{com}_{1,C})$.
- Run $(\mathsf{com}_{1,R}, K) = \mathsf{NIDI.V}(\tau, \Pi)$. If the verification fails, output $\perp$ and stop proceeding. Otherwise, follow the next steps.
- Depending on x, w sample ℓ_{ci} repetitions of $\Sigma.\mathsf{SampFirst}$. Namely, for $j \in [\ell_{ci}]$, compute $(a_{(j-1)N+1}, \ldots, a_{jN}) \leftarrow \Sigma.\mathsf{SampFirst}$.
- For $k \in [N\ell_{ci}]$, compute $\mathsf{com}_{2,C,k} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, a_k; s_k)$ for a freshly chosen s_k. Let $\mathsf{com}_{2,C} = \{\mathsf{com}_{2,C,k}\}_{k \in [N\ell_{ci}]}$.

- Run $\mathbf{e} = \mathcal{H}.\mathsf{Eval}(K, (x, \mathsf{com}_{1,C}, \mathsf{com}_{1,R}, \mathsf{com}_{2,C}))$.
- For $j \in [\ell_{ci}]$ determine Set_j, the set of commitments to be opened for j^{th} repetition, as per challenge bit e_j. Let Set be the union of these sets.
- Output $\mathsf{com}_{2,C}$ along with $\mathbf{e}$ and openings $z = \{a_k, s_k\}_{k \in \mathsf{Set}}$.

$\underline{\mathsf{ZKVerify}_2(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,P}, \mathsf{zk}_{2,P}, x)}$: Compute the following steps.

- Parse $\mathsf{zk}_{1,V} = \Pi$, $\mathsf{zk}_{1,P} = (\tau, \mathsf{com}_{1,C})$ and $\mathsf{zk}_{2,P} = (\mathsf{com}_{2,C}, \mathbf{e}, z = \{a_k, s_k\}_{k \in \mathsf{Set}})$.
- Compute $(\mathsf{com}_{1,R}, K) = \mathsf{NIDI.V}(\tau, \Pi)$ and check if $\mathbf{e} = \mathcal{H}.\mathsf{Eval}(K, (x, \mathsf{com}_{1,C}, \mathsf{com}_{1,R}, \mathsf{com}_{2,C}))$.
- Check if $z = \{a_k, s_k\}_{k \in \mathsf{Set}}$ are valid openings of $\{\mathsf{com}_{2,C,k}\}_{k \in \mathsf{Set}}$.
- Finally verify that $\{a_k\}_{k \in \mathsf{Set}}$ as a valid third message of ℓ_{ci} parallel repetition of Σ protocol according to $\mathbf{e}$ and instance x.
- Output 1 if every verification above succeeds, else output 0.

Remark 7. *We assume that the prover always outputs a valid first message $\mathsf{zk}_{1,P}$. This can be ensured as follows. If the first message is either not given out, or if one of τ and $\mathsf{com}_{1,C}$ is not valid, then we interpret $\tau = 0^{\ell_{\mathsf{NIDI}}}$ and $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(0^{\ell_\mu})$.*

We now argue various properties involved.

Completeness. Completeness is straightforward to argue and follows from perfect completeness of NIDI, perfect correctness of the SEE and perfect completeness of the Σ protocol.

9.3 Soundness

We now argue soundness. We first define the "soundness mode", and then argue all three properties.

Perfect Soundness Mode. Note that in order for a proof to verify $\mathsf{zk}_{1,P} = (\tau, \mathsf{com}_{1,C})$ needs to be verifiable. In particular, there must exist $\mathbf{b}_P$ such that $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_P)$ (where $\mathbf{b}_P$ is as chosen by the prover, or 0^{ℓ_μ} if the prover aborts, or outputs a non-well formed message). In the soundness game on the other hand, the verifier is honest and chooses $\mathbf{b}_V \leftarrow \{0,1\}^{\ell_\mu}$ and sets $\Pi = \mathsf{NIDI.P}(\mathcal{D}_{\mathbf{b}_V})$. We define the Perfect soundness mode to be the mode when $\mathbf{b}_P = \mathbf{b}_V$.

Lemma 13. *When $\mathbf{b}_P = \mathbf{b}_V$, then there does not exist an accepting proof of any $x \notin L$.*

Proof. When $\mathbf{b}_P = \mathbf{b}_V$, then consider any accepting proof say $(\mathsf{com}_{2,C}, \mathbf{e}, \{a_k, s_k\}_{\mathsf{Set}})$. Observe that $\mathbf{e} = \mathcal{H}.\mathsf{Eval}(K, x, \mathsf{com}_{1,C}, \mathsf{com}_{1,R}, \mathsf{com}_{2,C})$. Note that K is generated by using $\mathsf{FakeGen}$ algorithm with input the function f, which uses $\mathbf{b}_V$ and randomness $\mathbf{r}$ to decrypt all the commitments $\{\mathsf{com}_{2,C,k}\}_{k \in [\ell_{ci} N]}$. Let us say that this decryption results in $\{a'_k\}_{k \in [\ell_{ci} N]}$. Due to the correctness of the decryption/extraction of SEE, $a_k = a'_k$ for every $k \in \mathsf{Set}$ as the adversary opens it in the proof. Now, when $x \notin L$, the Σ protocol ensures that there is atmost one $\mathbf{e}_{bad}$ such that $\{a'_k\}_{k \in [\ell_{ci} N]}$ can lead to a valid proof. This $\mathbf{e}_{bad}$ is computed by the function f. The perfect CI property of $\mathcal{H}$ ensures that $\mathbf{e} \neq \mathbf{e}_{bad}$. On the other hand, if the adversary gives a valid proof for $\mathbf{e}$, then $\mathbf{e} = \mathbf{e}_{bad}$. This is a contradiction.

We now analyze the frequency of the perfect soundness mode.

Lemma 14. *For any honest polynomial time verifier V, and a cheating prover P^* in $\mathcal{C}_{sound}$, the soundness mode holds with probability at least $\Omega(2^{-\ell_\mu})$.*

Proof. We prove this by a simple reduction to the security of distributional indistinguishability of $\mathsf{NIDI.P}(\mathcal{D}_{\mathbf{b}_V})$. We show this using a hybrid argument.

Hybrid$_0$: In this hybrid, the challenger samples randomly $\mathbf{b}_V$ and outputs $\mathsf{zk}_{1,V}$ as $\mathsf{NIDI.P}(\mathcal{D}_{\mathbf{b}_V})$. Then, the prover outputs $\mathsf{zk}_{1,P} = \tau, \mathsf{com}_{1,C}$. The challenger outputs 1 if $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_V)$.

Hybrid$_1$: In this hybrid, the challenger samples randomly $\mathbf{b}_V$. It also samples randomly $\mathbf{b}'$ and outputs $\mathsf{zk}_{1,V}$ as $\mathsf{NIDI.P}(\mathcal{D}_{\mathbf{b}'})$. Then, the prover outputs $\mathsf{zk}_{1,P} = \tau, \mathsf{com}_{1,C}$. The challenger outputs 1 if $\mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_V)$.

In order to prove the claim, our first observation is that soundness mode holds when **Hybrid**$_0$ outputs 1. Second, observe that the probability that **Hybrid**$_1$ outputs 1 is exactly $2^{-\ell_\mu}$. Our claim follows from the fact that for any adversary $\mathcal{A} \in \mathcal{C}_{sound}$, it holds that the these to hybrids are indistinguishable with advantage bounded by $\epsilon_{\mathsf{NIDI},DI}$. This is due to the security of NIDI and the indistinguishability property of the distribution $\mathcal{D}_{\mathbf{b}'}$ for a random $\mathbf{b}'$ from $\mathcal{D}_{\mathbf{b}_V}$. Thus, the claim holds. □

We now argue indistinguishability of the soundness mode property.

Lemma 15. *The construction of* zk *satisfies* $(\mathcal{C}_{sound}, \epsilon_{sound,2})$ *indistinguishability of soundness property with* $\epsilon_{sound,2} = O(\epsilon_{\mathsf{NIDI},DI} 2^{\ell_\mu})$.

Proof. Let P^* be a cheating prover in $\mathcal{C}_{sound}$, and V be an honest verifier in the soundness experiment. Let E be the distribution of the transcript. Let E_1 denote the distribution of transcript when the soundness mode holds and E_0 denote the distribution of transcript when the soundness mode does not hold. Let $\mathcal{A}$ be any adversary in $\mathcal{C}_{sound}$. Then, we want to bound the following probability.

$$p = \left| \Pr[\mathcal{A}(e) = 1 | e \leftarrow \mathsf{E}_0] - \Pr[\mathcal{A}(e) = 1 | e \leftarrow \mathsf{E}_1] \right|$$

Every instance of e consists of $\mathsf{zk}_{1,V}$ and $\mathsf{zk}_{1,P}$ output by the cheating prover. Let S denote the set of elements in the range of $\mathsf{Com}_{1,C}(\star)$. There are exactly 2^{ℓ_μ} elements in this set. For every $s \in S$, we define $\mathsf{E}_{0,s}$ to be the collection of transcripts in E_0 where the verifier submits s as $\mathsf{com}_{1,C}$. Likewise, we define $\mathsf{E}_{1,s}$ to be the collection of transcripts in E_1 where the verifier submits s as $\mathsf{com}_{1,C}$. Thus, due to triangle inequality we have that:

$$p < \sum_{s \in S} \left| \Pr[\mathcal{A}(e) = 1 \wedge \mathsf{zk}_{1,P} = s | e \leftarrow \mathsf{E}_{0,s}] - \Pr[\mathcal{A}(e) = 1 \wedge \mathsf{zk}_{1,P} = s | e \leftarrow \mathsf{E}_{1,s}] \right|.$$

We will prove that for every $s \in S$,

$$\left| \Pr[\mathcal{A}(e) = 1 \wedge \mathsf{zk}_{1,P} = s | e \leftarrow \mathsf{E}_{0,s}] - \Pr[\mathcal{A}(e) = 1 \wedge \mathsf{zk}_{1,P} = s | e \leftarrow \mathsf{E}_{1,s}] \right| < O(\epsilon_{\mathsf{NIDI},DI}).$$

This will prove the claim. We show this as follows. Assume towards contradiction that there exist s^* such that.

$$\left| \Pr[\mathcal{A}(e) = 1 \wedge \mathsf{zk}_{1,P} = s^* | e \leftarrow \mathsf{E}_{0,s^*}] - \Pr[\mathcal{A}(e) = 1 \wedge \mathsf{zk}_{1,P} = s^* | e \leftarrow \mathsf{E}_{1,s^*}] \right| = \epsilon_1.$$

We will use this to attack the indistinguishability of NIDI with the probability ϵ_1. We will show that if this happens, then we can build a reduction using $\mathcal{A}$ that is also in $\mathcal{C}_{sound}$, that distinguishes $\mathsf{NIDI.P}(\mathcal{D}_{\mathbf{b}_0})$ from $\mathsf{NIDI.P}(\mathcal{D}_{\mathbf{b}_1})$ with an advantage ϵ_1 where $\mathbf{b}_1 = \mathsf{Com}_{1,C}^{-1}(s^*)$ and $\mathbf{b}_0$ is uniformly sampled from $\{0,1\}^{\ell_\mu} \setminus \mathsf{Com}_{1,C}^{-1}(s^*)$. The reduction works as follows.

- Obtain the challenge NIDI proof Π.
- Send Π to the prover P^*. Prover outputs τ, s. If $s = s^*$, then output $\mathcal{A}(\Pi, \tau, s)$, otherwise output $\perp$.

If Π is generated using $\mathbf{b}_0$, then, the transcript is not in the soundenss mode when P^* outputs s^*, where as if Π is generated using $\mathbf{b}_1$, then the transcript is in soundness mode when P^* outputs s^*. Observe that the advantage of the reduction is exactly equal to ϵ_1. Therefore, $\epsilon_1 \leq \epsilon_{\mathsf{NIDI},DI}$ as per the parameters set, which is a contradiction. □

9.4 Zero-Knowledge

We now argue the zero-knowledge properties of the protocol. We begin by describing our simulator, $\mathsf{zk}.\mathcal{S}$ and then argue why the security holds. The simulator will run in time polynomial in 2^{λ^ρ}.
$\underline{\mathsf{zk}.\mathcal{S}(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,P}, x)}$: Compute the following steps.

- Parse $\mathsf{zk}_{1,V} = \Pi$ and $\mathsf{zk}_{1,P} = (\tau, \mathsf{com}_{1,C})$.
- Run $(\mathsf{com}_{1,R}, K) \leftarrow \mathsf{NIDI.V}(\tau, \Pi)$. If the verification fails, output $\perp$. Else, continue.
- Compute $\mathsf{com}_{2,C}$ by setting $\mathsf{com}_{2,C,k} = \mathsf{Com}_{2,C}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, 0; r'_k)$ for $k \in [N \cdot \ell_{ci}]$. Then set $\mathsf{com}_{2,C} = \{\mathsf{com}_{2,C,k}\}$.
- Run $\mathbf{e} \leftarrow \mathcal{H}.\mathsf{Eval}(K, (x, \mathsf{com}_{1,C}, \mathsf{com}_{1,R}, \mathsf{com}_{2,C}))$.
- Run the simulator of Σ protocol on input x and $\mathbf{e}$, and receive $\{a_k\}_{k \in \mathsf{Set}}$.
- Equivocate $\mathsf{com}_{2,C,k}$ for $k \in \mathsf{Set}$. That is, compute $\mathsf{SEE}.\mathcal{S}(\mathsf{com}_{1,R}, \mathsf{com}_{1,C}, r'_k, a_k) \to s_k$ for $k \in \mathsf{Set}$. Output $\perp$ if the equivocation fails.
- Set $z = \{a_k, s_k\}_{k \in \mathsf{Set}}$ and output $\mathsf{zk}_{2,P} = (\mathsf{com}_{2,C}, \mathbf{e}, z)$.

We now argue why the security property holds. Our first observation is that there is a simple criteria when the distribution $\mathsf{zk}.\mathcal{S}(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,P}, x)$ is identical to $\mathsf{ZKProve}_2(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,P}, x, w)$. In the zero-knowledge game $\mathsf{zk}_{1,P}$ is generated honestly containing $(\tau, \mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_P))$ for a randomly chosen $\mathbf{b}_P$. Now consider $\mathsf{zk}_{1,V}$ consisting of a NIDI proof Π. Let $(\mathsf{com}_{1,R}, K)\mathsf{NIDI.V}(\tau, \Pi)$. Since the cheating verifier is of depth $\mathsf{poly}(\lambda)$ and size polynomial in 2^λ and NIDI is sound against such adversaries, it holds that one of the scenarios must happen:

- Either $\mathsf{NIDI.V}$ outputs $\perp$,
- Or, if $\mathsf{NIDI.V}$ outputs $\mathsf{com}_{1,R}, K$, it must happen that $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_V)$ for some $\mathbf{b}_V$, or else the verifier violates soundness which is computationally hard.

We will show first that when $\mathsf{Com}_{1,C}(\mathbf{b}_V) \neq \mathsf{Com}_{1,C}(\mathbf{b}_P)$ of if V outputs $\perp$, then the simulator above produces an identical distribution to the honest proving algorithm. When, this does not happen, the verifier must either:

- Break soundness of NIDI, or,

- Force $\mathsf{zk}_{1,P} = \mathsf{Com}_{1,C}(\mathbf{b}_V)$ which is hard due to the security of SEE.

This will finish the analysis.

Lemma 16. *Let (x, w) be a valid instance-witness pair. Let $\mathsf{zk}_{1,P} = (\tau, \mathsf{com}_{1,C} = \mathsf{Com}_{1,C}(\mathbf{b}_P))$. Let $\mathsf{zk}_{1,V} = \Pi$ such that either:*

- $\mathsf{NIDI.V}(\tau, \Pi) = \perp$, *or,*
- $\mathsf{NIDI.V}(\tau, \Pi) = \mathsf{com}_{1,R}, K$, *where $\mathsf{com}_{1,R} = \mathsf{Com}_{1,R}(\mathbf{b}_V)$. for $\mathbf{b}_V \neq \mathbf{b}_P$.*

Then, $(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,P}, \mathsf{zk}_{2,P} = \mathsf{zk}.\mathcal{S}(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,V}, x))$ is identically distributed to $(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,P}, \mathsf{zk}_{2,P} = \mathsf{ZKProve}_2(\mathsf{zk}_{1,V}, \mathsf{zk}_{1,V}, x, w))$ where the randomness is only over the generation of proof $\mathsf{zk}_{2,P}$.

The proof of this immediate. If $\mathsf{NIDI.V}(\tau, \Pi) = \perp$ then, both algorithms output $\perp$, which is identically distributed. In case of the second criteria, the proof is also immediate. It follows from equivocation property of the commitment scheme and honest verifier zero-knowledge of SEE. We show it by three hybrids where the first hybrid corresponds to the actual proof, and the last hybrid corresponds to the simulator.
Hybrid$_0$: In this hybrid, run as in the honest algorithm to compute $\mathsf{zk}_{2,P}$: sample $\{a_k\}_{k\in[N\ell_{ci}]}$ as in the Σ protocol and then commit them to compute $\mathsf{com}_{2,C}$. Apply $\mathcal{H}$ on $\mathsf{com}_{2,C}$ to derive $\mathbf{e}$, and then open the commitments to $\{a_k\}_{k\in\mathsf{Set}}$ honestly.
Hybrid$_1$: In this hybrid, we make the following change to generate $\mathsf{zk}_{2,P}$: we sample $\{a_k\}_{k\in[N\ell_{ci}]}$ as in the Σ protocol but then commit $0's$ instead of $\{a_k\}$ to compute $\mathsf{com}_{2,C}$. Then, we apply $\mathcal{H}$ on $\mathsf{com}_{2,C}$ to derive $\mathbf{e}$. At the opening time, we open these commitments by using $\mathsf{SEE}.\mathcal{S}$ to equivocate these commitments to open to $\{a_k\}_{k\in\mathsf{Set}}$.

Note that since $\mathsf{com}_{1,C} \neq \mathsf{com}_{1,R}$, the distribution of these two hybrids are identical due to equivocation property of SEE.

Hybrid$_2$: In this hybrid, we make the following change to generate $\mathsf{zk}_{2,P}$: we generate $\mathsf{com}_{2,C}$ by committing to $0's$. Then, we apply $\mathcal{H}$ on $\mathsf{com}_{2,C}$ to derive $\mathbf{e}$. At the opening time, we first sample $\{a_k\}_{k\in\mathsf{Set}}$ using the honest verifier simulator of Σ protocol, and then open the commitments of 0 by using $\mathsf{SEE}.\mathcal{S}$ to $\{a_k\}_{k\in\mathsf{Set}}$.
These hybrid corresponds to $\mathsf{zk}.\mathcal{S}$. Note that due to the security of the Σ protocol, the last two hybrids are identical. □

The lemma above solves our problems completely, except that we must ensure that the conditions for when the distributions are not identical outlined above does not happen in the zero knowledge security game.

We show this using hybrids. The first hybrid corresponds to the case of honest experiment. The last hybrid corresponds to the simulated experiment.

Hybrid$_0$: This hybrid corresponds to the experiment where the responses are done using the honest $\mathsf{ZKProve}_2$ algorithm. Throughout, parse $\mathsf{zk}_{1,P} = (\tau, \mathsf{com}_{1,C})$.

Hybrid$_1$: This hybrid is the same as before, except that we abort if the cheating verifier queries $(x_i, w_i, \mathsf{zk}_{1,V,i} = \Pi_i)$ such that $\mathsf{NIDI.V}(\tau, \Pi_i) = \mathsf{com}_{1,R,i}, K_i$ where $\mathsf{com}_{1,R,i} = \mathsf{Com}_{1,R}(\mathbf{b}_{V,i})$ such that $\mathsf{Com}_{1,C}(\mathbf{b}_{V,i}) = \mathsf{com}_{1,C}$.

Note that the above two hybrids are statistically close. This is because V^* is an adversary of polynomially bounded depth and size polynomial in 2^λ. The commitment scheme SEE ensures that any adversary of polynomial depth, and size bounded by $2^{\lambda_{\mathsf{com}}} \gg 2^\lambda$ cannot produce $\mathsf{com}_{1,R}$

with this property with advantage more than $2^{-\lambda_{\mathsf{com}}} \ll 2^{-\lambda}$. Thus, probability of abort is less than $2^{-\lambda_{\mathsf{com}}}$. We also make a note that the challenger for this hybrid can be run in time polynomial in $2^{t_{\mathsf{com}}} = 2^{\lambda^\rho}$. This is to break open $\mathsf{com}_{1,R}$.

Hybrid$_2$: This hybrid is the same as before, except that we abort if the cheating verifier queries $(x_i, w_i, \mathsf{zk}_{1,V,i} = \Pi_i)$ such that $\mathsf{NIDI.V}(\tau, \Pi_i) = (\mathsf{com}_{1,R,i}, K_i)$ where $\mathsf{com}_{1,R,i} \neq \mathsf{Com}_{1,R}(\mathbf{b}_{V,i}; \mathbf{r}_i)$ or $K_i = \mathcal{H}.\mathsf{FakeGen}(f[\mathsf{com}_{1,R,i}, \mathbf{b}_{V_i}, \mathbf{r}_i])$.

Note that the above two hybrids are statistically close. This is because, if there is a cheating verifier V^* of depth $\mathsf{poly}(\lambda)$ and size polynomial in 2^λ that produces distinguishable hybrids, then we can build a reduction of size polynomial in 2^λ that violates soundness of NIDI. The reduction responds to the queries as in **Hybrid**$_1$. It also needs to run to respond to the queries to figure out aborting conditions as in **Hybrid**$_1$. It can do so, by brute-force opening of $\mathsf{com}_{1,R,i}$, which can be done by a circuit of size $2^{t_{\mathsf{com}} = \lambda^\rho}$. Since NIDI is sound against adversaries of size polynomial $2^{\lambda_{\mathsf{NIDI}}} \gg 2^\lambda$ with advantage less than $2^{-\lambda}$, these two hybrids are statistically close (unless the reduction wins in the soundness game).

Hybrid$_3$: This hybrid is the same as before, except that we simulate $\mathsf{zk}_{2,P}$ responses.

These hybrids are identical due to Lemma 16.

References

[ABG+21] Amit Agarwal, James Bartusek, Vipul Goyal, Dakshita Khurana, and Giulio Malavolta. Two-round maliciously secure computation with super-polynomial simulation. *IACR Cryptol. ePrint Arch.*, page 1230, 2021. To appear in TCC 2021.

[AJJM21] Prabhanjan Ananth, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Unbounded multi-party computation from learning with errors. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 754–781. Springer, Heidelberg, October 2021.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, Heidelberg, April 2012.

[AJW11] Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. Cryptology ePrint Archive, Report 2011/613, 2011. https://eprint.iacr.org/2011/613.

[BDD+20] Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. CRAFT: composable randomness and almost fairness from time. *IACR Cryptol. ePrint Arch.*, page 784, 2020.

[BDD+21] Carsten Baum, Bernardo David, Rafael Dowsley, Jesper Buus Nielsen, and Sabine Oechsner. TARDIS: A foundation of time-lock puzzles in UC. In *Advances in Cryptology - EUROCRYPT*, pages 429–459, 2021.

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. In *Advances in Cryptology - EUROCRYPT*, pages 79–109, 2020.

[BFJ+20] Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 642–667. Springer, Heidelberg, May 2020.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, 2012.

[BGI+14] Amos Beimel, Ariel Gabizon, Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, and Anat Paskin-Cherniavsky. Non-interactive secure multiparty computation. In *Advances in Cryptology - CRYPTO*, pages 387–404, 2014.

[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 *Advances in Cryptology - ASIACRYPT*, pages 275–303, 2017.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, ITCS*, pages 345–356, 2016.

[BGJ+17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 743–775. Springer, Heidelberg, November 2017.

[BGMM20] James Bartusek, Sanjam Garg, Daniel Masny, and Pratyay Mukherjee. Reusable two-round MPC from DDH. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 320–348. Springer, Heidelberg, November 2020.

[BGSZ21] James Bartusek, Sanjam Garg, Akshayaram Srinivasan, and Yinuo Zhang. Reusable two-round MPC from LPN. *IACR Cryptol. ePrint Arch.*, page 316, 2021.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

[BJKL21] Fabrice Benhamouda, Aayush Jain, Ilan Komargodski, and Huijia Lin. Multiparty reusable non-interactive secure computation from LWE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 724–753. Springer, Heidelberg, October 2021.

[BKP18] Nir Bitansky, Yael Tauman Kalai, and Omer Paneth. Multi-collision resistance: a paradigm for keyless hash functions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 671–684. ACM, 2018.

[BL18] Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In *Theory of Cryptography - 16th International Conference, TCC*, pages 209–234, 2018.

[BL20] Fabrice Benhamouda and Huijia Lin. Mr NISC: Multiparty reusable non-interactive secure computation. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 349–378. Springer, Heidelberg, November 2020.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In Harriet Ortiz, editor, *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513. ACM, 1990.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In *Advances in Cryptology - CRYPTO*, pages 236–254, 2000.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019.

[DJMW12] Yevgeniy Dodis, Abhishek Jain, Tal Moran, and Daniel Wichs. Counterexamples to hardness amplification beyond negligible. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC*, pages 476–493. Springer, 2012.

[DKP21] Dana Dachman-Soled, Ilan Komargodski, and Rafael Pass. Non-malleable codes for bounded parallel-time tampering. In *Advances in Cryptology - CRYPTO*, pages 535–565, 2021.

[DQV+21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct lwe sampling, random polynomials, and obfuscation. *IACR Cryptol. ePrint Arch.*, page 1226, 2021.

[EFKP20] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Non-malleable time-lock puzzles and applications. *IACR Cryptol. ePrint Arch.*, page 779, 2020.

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Symposium on Theory of Computing Conference, STOC*, pages 467–476, 2013.

[GJJM20] Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Statistical zaps and new oblivious transfer protocols. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 668–699. Springer, Heidelberg, May 2020.

[GMPY11] Juan A. Garay, Philip D. MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. *J. Cryptol.*, 24(4):615–658, 2011.

[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. *Journal of Cryptology*, 7(1):1–32, December 1994.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 736–749, 2021.

[JKKR17] Abhishek Jain, Yael Tauman Kalai, Dakshita Khurana, and Ron Rothblum. Distinguisher-dependent simulation in two rounds and its applications. In *Advances in Cryptology - CRYPTO*, pages 158–189, 2017.

[JLS21a] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, pages 60–73, 2021.

[JLS21b] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation without lattices, 2021. Unpublished manuscript.

[Khu21] Dakshita Khurana. Non-interactive distributional indistinguishability (NIDI) and non-malleable commitments. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 186–215. Springer, Heidelberg, October 2021.

[KK19] Yael Tauman Kalai and Dakshita Khurana. Non-interactive non-malleability from quantum supremacy. In *Advances in Cryptology - CRYPTO*, pages 552–582, 2019.

[KKS18] Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Statistical witness indistinguishability (and more) in two messages. In *Advances in Cryptology - EUROCRYPT*, pages 34–65, 2018.

[KLX20] Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC*, pages 390–413, 2020.

[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.

[LPS20] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. *SIAM J. Comput.*, 49(4), 2020.

[MPP20] Andrew Morgan, Rafael Pass, and Antigoni Polychroniadou. Succinct non-interactive secure computation. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 216–245. Springer, Heidelberg, May 2020.

[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.

[Pin03] Benny Pinkas. Fair secure two-party computation. In *Advances in Cryptology - EUROCRYPT*, pages 87–105, 2003.

[PPV08] Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *Advances in Cryptology - CRYPTO*, pages 57–74, 2008.

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, STOC*, pages 242–251. ACM, 2004.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019*, 2019.

[RS20] Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. In *Advances in Cryptology - CRYPTO*, pages 481–509, 2020.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto, 1996. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA.

[SW21] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *SIAM J. Comput.*, 50(3):857–908, 2021.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In *Advances in Cryptology - EUROCRYPT*, pages 127–156, 2021.

[Yao86a] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

[Yao86b] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.