Concurrent Secure Computation via Non-Black Box Simulation

Vipul Goyal
Microsoft Research India
vipul@microsoft.com

Divya Gupta
University of California, Los Angeles
and Center for Encrypted Functionalities
divyag@cs.ucla.edu

Amit Sahai
University of California, Los Angeles
and Center for Encrypted Functionalities
sahai@cs.ucla.edu

November 13, 2015

Abstract

Recently, Goyal (STOC'13) proposed a new non-black-box simulation techniques for fully concurrent zero knowledge with straight-line simulation. Unfortunately, so far this technique is limited to the setting of concurrent zero knowledge. The goal of this paper is to study what can be achieved in the setting of concurrent secure computation using non-black-box simulation techniques, building upon the work of Goyal. The main contribution of our work is a secure computation protocol in the fully concurrent setting with a straight-line simulator, that allows us to achieve several new results:

- We give first positive results for concurrent blind signatures and verifiable random functions in the plain model *as per the ideal/real world security definition*. Our positive result is somewhat surprising in light of the impossibility result of Lindell (STOC'03) for black-box simulation. We circumvent this impossibility using non-black-box simulation. This gives us a quite natural example of a functionality in concurrent setting which is impossible to realize using black-box simulation but can be securely realized using non-black-box simulation.
- Moreover, we expand the class of realizable functionalities in the concurrent setting. Our main theorem is a positive result for concurrent secure computation as long as the ideal world satisfies the *bounded pseudo-entropy condition* (BPC) of Goyal (FOCS'12). The BPC requires that in the ideal world experiment, the total amount of information learnt by the adversary (via calls to the ideal functionality) should have "bounded pseudoentropy".
- We also improve the round complexity of protocols in the single-input setting of Goyal (FOCS'12) both qualitatively and quantitatively. In Goyal's work, the number of rounds depended on the length of honest party inputs. In our protocol, the round complexity depends only on the security parameter, and is completely independent of the length of the honest party inputs.

Our results are based on a non-black-box simulation technique using a new language (which allows the simulator to commit to an Oracle program that can access information with bounded pseudoentropy), and a simulation-sound version of the concurrent zero-knowledge protocol of Goyal (STOC'13). We assume the existence of collision resistant hash functions and constant round semi-honest oblivious transfer.

1 Introduction

Secure computation protocols enable a set of mutually distrustful parties to securely perform a task by interacting with each other. Traditional security notions for secure computation [Yao86, GMW87] were defined for the *stand-alone setting* where security holds only if a single protocol session is executed in isolation. In today's connected world (and especially over internet), many instances of these protocols may be executing concurrently. In such a scenario, a protocol that is secure in the classical stand-alone setting may become completely insecure [Lin03b, BPS06]. Ambitious efforts have been made to generalize the results for the stand-alone setting, starting with concurrently-secure zero-knowledge protocols [DNS98, RK99, CKPR01, KP01, PRS02].

However, in the plain model, the effort to go beyond the zero-knowledge functionality were, unfortunately, less than fully satisfactory. In fact, for the plain model far reaching unconditional impossibility results were shown in a series of works [CKL03, Lin03b, Lin08, BPS06, Goy12, AGJ$^+$12, GKOV12]. Two notable exceptions giving positive results in the plain model are the works on *bounded* concurrency [Lin03a, PR03, Pas04] (where there is an a-priori fixed bound on the total number of concurrent sessions in the system and the protocol in turn can depend on this bound), and, the positive results for a large class of functionalities in the so called "single input" setting [Goy12]. In this setting, there is a server interacting with multiple clients concurrently with the restriction that the server (if honest) is required to use the *same* input in all sessions. There is a large body of literature on getting concurrently secure computation in weaker models such as using a super-polynomial time simulator, or a trusted setup. A short survey of these works is given later in this section. We emphasize that in this work, we are interested in concurrently secure computation protocols with no trusted set up assumptions where the security holds according to standard ideal/real paradigm.

An intriguing functionality that cannot be realized in the fully concurrent setting by these results is blind signatures in the plain model. The blind signature functionality, introduced by [Cha82], allows users to obtain unforgeable signatures on messages of their choice without revealing the message being signed to the signer (blindness property). The question of whether a concurrently-secure protocol for this functionality can be constructed as per the ideal/real model simulation paradigm has been open so far. Moreover, given the impossibility result for concurrent blind signatures for black box simulation by Lindell [Lin03b], it is clear that we need to use non-black box techniques. Until recently, no non-black box technique was known which applies to full concurrency with polynomial time simulation. However, Goyal [Goy13] recently proposed new non-black box simulation techniques for (fully) concurrent zero-knowledge with straight line simulation. Unfortunately, the result of Goyal is limited to the setting of concurrent zero-knowledge. We ask the question: *Can we construct non-box black techniques for (fully) concurrent secure computation, building upon the work of Goyal [Goy13]?*

Our Contributions. The main contribution of our work is a secure computation protocol in the fully concurrent setting with a straight-line simulator, that allows us to achieve several new results. In short, we expand the class of realizable functionalities in the concurrent setting and give the first positive results for concurrent blind signatures and verifiable random functions in the plain model *as per the ideal/real world security definition*. Moreover, the round complexity of our protocol depends only on the security parameter and hence, improves the round complexity of [Goy12] both qualitatively and quantitatively. Finally, our work can be seen as a *unifying framework*, which

essentially subsumes all the previous work on positive results for concurrent secure computation achieving polynomial time simulation based security in the plain model. For detailed description of our results, see Section 1.1.

Other models. In order to circumvent the above mentioned impossibility results in the plain model, there has been quite some work studying various trust assumptions such as common reference string (CRS) model and tamper proof hardware tokens [CLOS02, BCNP04, Kat07]. Another interesting line of work has studied weaker security definitions [GM00, Pas03, PS04, MPR06] while still remaining in the plain model, and most notably obtains positive results in models like super polynomial time simulation [PS04, BS05, CLP10, GGJS12] and input indistinguishable security [MPR06, GGJS12].

Note that these trust assumptions and these relaxed notions of security are sometimes restrictive and are not applicable to many situations. We again emphasize that the focus of this work is concurrent secure computation in the plain model achieving *polynomial* time simulation. In the plain model, there are point to point authenticated channels between the parties, but there is no global trusted third party.

What goes wrong in concurrent setting in plain model? A well established approach to constructing secure computation protocols is to use the GMW compiler: take a semi-honest secure computation protocol and "compile" it with zero-knowledge arguments. The natural starting point in the concurrent setting is to follow the same principles: somehow compile a semi-honest secure computation protocol with a *concurrent* zero-knowledge protocol (actually compile with concurrent *non-malleable* zero-knowledge [BPS06]). Does such an approach (or minor variants) already give us protocols secure according to the standard ideal/real world definition in the plain model?

There is a fundamental problem with this approach which poses a key *bottleneck* in a number of previous works (see [GS09, GJO10, GM11, GGJS12, Goy12, GGJ13]). All known concurrent zero-knowledge simulators in the fully concurrent setting work by rewinding the adversarial parties. Such an approach is highly problematic for secure computation in the concurrent setting, where the adversary controls the scheduling of the messages of different sessions. For instance, consider the following scenario: Due to nesting of sessions by the adversary, a rewinding based simulator may need to execute some sessions more than once. Since the adversary can choose a different input in each execution (e.g. based on transcript so far), the simulator would have to query the ideal functionality for than once. However, for any session, the simulator is allowed at most one query! Indeed, such problems are rather inherent as indicated by various impossibility results [Lin08, BPS06].

Trying to solve this bottleneck of "handling extra queries" in various ways has inspired a number of different works which revolve around a unified theme: first construct a protocol where the simulator requires multiple queries per session in the ideal world, and then, somehow manage to either eliminate or answer these extra queries by exploiting some property of the specific setting in question. Examples of these include Resettable and Stateless computation [GS09, GM11], Multiple Ideal Query model [GJO10, GJ13, GGJ13], Single-Input setting [Goy12], Leaky Ideal Query model [GGJ13], etc[1].

Indeed, as is natural to expect, there are limitations on how much one can achieve using the above paradigm of constructing protocols. A very natural question that arises is *whether there*

[1]For a detailed survey of these works, see Appendix G

exists a different approach which allows us to construct concurrent secure computation protocols in the plain model without the need of additional output queries? Moreover, if such a different approach does exist, we know that due to impossibility results[CKL03, Lin03b, Lin08, BPS06, Goy12, AGJ$^+$12, GKOV12], there will be some limitations on the scope of its applicability. This leads to some more natural questions. *What all can we achieve using this approach? In particular, can we expand the class of realizable functionalities in the concurrent setting? Can we improve the parameters (e.g. round complexity) of the protocols which exist in the plain model?*

1.1 Our Results

The key contribution of this work is a new way of approaching the problem of concurrent secure computation in the plain model facilitated by recent advances in concurrent non-black box simulation [Goy13]. We give a protocol with non-black box and straightline simulator. Since, very informally, our simulator does not rely on rewinding at all, we are able to avoid the key bottleneck of additional output queries to the ideal functionality during the rewinds.

However, our simulator has to overcome a number of additional obstacles not present in [Goy13]. Note that unlike secure computation, an adversary in concurrent zero-knowledge does not receive any outputs. Dealing with the outputs given to the adversary in each session is a key difficulty we have to overcome. In particular, one might think that a straightline simulator for concurrent zero-knowledge should give a concurrently secure computation protocol trivially for all functionalities and in particular for concurrently secure oblivious transfer. Note that this *cannot* be true given unconditional impossibility results for oblivious transfer. For more on such technical hurdles, please refer to the technical overview (Section 1.2).

Informally stated, our main theorem is a general positive result for concurrent secure computation as long as the ideal world satisfies our so called *bounded pseudo-entropy condition* (BPC). Very informally, the bounded pseudoentropy condition requires that in the ideal world experiment, the total amount of information learnt by the adversary (via calls to the trusted party) should have "bounded pseudoentropy". The origin of the bounded pseudoentropy condition comes from a conjecture of Goyal [Goy12]. More precisely, the bounded pseudoentropy condition says the following:

Definition 1 (Bounded Pseudoentropy Condition (BPC)) *An ideal world experiment satisfies bounded pseudoentropy condition if there exists $B \in \mathbb{N}$ and a PPT algorithm $\mathcal{T}$ such that for all $m = m(n)$ concurrent sessions, for all adversarial input vectors $\vec{I}$ (where an element of the vector represents the input of the adversary in that session), there exists a set S of possible output vectors such that the following conditions are satisfied*

- *All valid output vectors corresponding to the input vector $\vec{I}$ of the adversary are contained in S. Observe that for a given $\vec{I}$, for different honest party input vectors, the output vectors may be different. We require that any such output vector be contained in S. Furthermore, $|S| \leq 2^B$.*
- *For every $\vec{O} \in S$, $\mathcal{T}(\vec{I}, \vec{O}) = 1$, and for every $\vec{O} \notin S$, $\mathcal{T}(\vec{I}, \vec{O}) = 0$. That is, the set S is efficiently recognizable.*

Intuitively, this condition says the following: The adversary might be scheduling an unbounded polynomial number of sessions and gaining information from each of the outputs obtained. However for any vector of adversarial inputs, the number of possible output vectors is bounded (and hence

so is the information that adversary learns). Further note that this condition places a restriction only on the ideal world experiment, which consists of the functionality being computed and the honest party inputs. There is no restriction on the ideal world adversary, which may follow any (possibly unbounded state) polynomial time strategy.

It can be seen that in concurrent zero-knowledge, as well as, in the bounded concurrency setting, the BPC is satisfied. Also note that the class of ideal worlds which satisfy BPC is significantly more general compared to the single input setting of [Goy12]. For a formal proof of this claim, refer to Appendix C. In our work, we prove the following main theorem.

Theorem 1 *Assume the existence of collision resistant hash functions and constant-round semi-honest oblivious transfer. If the ideal world for the functionality $\mathcal{F}$ satisfies the bounded pseudoentropy condition in Definition 1, then for any constant ϵ, there exists a $O(n^{\epsilon})$ round real world protocol Π which securely realizes the ideal world for functionality $\mathcal{F}$.*

To understand the power of our result, a positive result for all ideal worlds satisfying BPC allows us to get the following "concrete" results:

- **Resolving the bounded pseudoentropy conjecture.** Goyal [Goy12] considered the so called "single input setting" and obtained a positive result for many functionalities in the plain model. Goyal further left open the so called bounded pseudoentropy conjecture which if resolved would give a more general and cleaner result (see [Goy12] for the exact statement).

  Our BPC is inspired from this conjecture (and can be seen as one way of formalizing it). Thus, Theorem 1 allows us to resolve the bounded pseudoentropy conjecture in the positive. Our positive result for the BPC subsumes most known positive results for concurrent secure computation in the plain model such as for zero-knowledge [RK99, KP01, PRS02], bounded concurrent computation [Lin03a, PR03, Pas04], and the positive results in the single input setting [Goy12].

- **Improving the round complexity of protocols in the single input setting.** The round complexity of the construction of Goyal [Goy12] in the single input setting was a large polynomial depending not only upon the security parameter but also on the length of the input and the nature of the functionality. For example, for concurrent private information retrieval, the round complexity would depend multiplicatively of the number of bits in the database and the security parameter. Our construction only has n^{ϵ} rounds, where n is the security parameter. Therefore, we obtain a significant qualitative improvement in the round complexity for protocols in the single input setting.

- **Expanding the class of realizable functionalities, and, getting blind signatures.** The blind signature functionality is an interesting case in the paradigm of secure computation both from theoretical as well as practical standpoints. The question of whether concurrent blind signatures (secure as per the ideal/real model simulation paradigm) exist is currently unresolved. Lindell [Lin03a, Lin08] showed an impossibility result for concurrent blind signature based on *black-box simulation*. This result has also been used as a motivation to resort to weaker security notions (such as game based security) or setup assumptions in various subsequent works (see e.g., [Fis06, Oka06, KZ06, HKKL07, GRS^{+}11, GG14]). We show that a positive result for BPC directly implies a construction of concurrent blind signatures *secure in the plain model as per the standard ideal/real world security notion.* Prior to our work, the

only known construction of concurrently secure blind signatures was according to the weaker game based security notion due to Hazay et al. [HKKL07].

This implies that concurrent blind signatures is a "natural" example of a functionality which is impossible to realize using black-box simulation but can be securely realized using non-black box simulation in the concurrent setting.[2] The only previous such example known [GM11] was for a reactive (and arguably rather contrived) functionality. Another concrete (and related) example of a new functionality that can be directly realized using our techniques is that of a secure verifiable random function.

It would also be interesting to see what our approach yields in the plain model for different settings and security notions where the previous rewinding based approach has been useful (such as resettable computation, super-polynomial simulation, etc). We leave that as future work.

1.2 Our Techniques

Our protocol and analysis for the concurrent secure computation is admittedly quite complex and we face a number of hurdles on the way. Below, we try to sketch the main difficulties and our ideas to circumvent them at a high level.

To construct concurrent secure computation, we roughly follow the [GMW87] strategy of first constructing an appropriate zero-knowledge protocol, and then "somehow compiling" a semi honest secure computation protocol using that. In our concurrent setting, in order to avoid the multiple output queries per session, we need a concurrently secure protocol for zero-knowledge with a *straightline* simulator. Recently, the first such protocol was given by Goyal [Goy13] based on non-black box techniques[3].

Another property of the zero-knowledge protocol which is crucial for compilation is *simulation-soundness*. Our first (and arguably smaller) technical hurdle is to construct a simulation-sound version of Goyal's protocol. This is necessary because the simulator would rely on the soundness of the proofs given by the adversary while simulating the proofs where it is acting as the prover. Another issue is that in our protocol for concurrent secure computation, the adversary is allowed to choose the statement proved till a very late stage in the protocol. Hence, we need simulation-soundness to hold even when the statements to prove are being chosen adaptively by the adversary. We note that this issue is somewhat subtle to deal with. Our construction of simulation-sound concurrent zero-knowledge relies on the following ingredients: Goyal's concurrent simulation strategy, a robust non-malleable commitment scheme [LP09], and a special language to be used in the universal arguments. The final construction along with a description of the main ideas is given in Section 3.

The next (and arguably bigger) difficulty is the following. In secure computation, the adversary receives an output in each session (this is unlike the case of zero-knowledge). It turns that that it is not clear how to handle these outputs while performing non-black box simulation. Note that

[2] Previous separations between the power of black-box and non-black box simulation are known only if we place additional constraints on the design of the real world protocol (e.g., it should be public coin, or constant rounds, etc.)

[3] Before this, all the (fully) concurrent zero-knowledge protocols were based on rewinding techniques, while, the construction of [Bar01] (which had a non-rewinding simulator) worked only in the bounded concurrent setting. The main result in [Goy13] was the first public-coin concurrent zero-knowledge protocol where the non-rewinding nature of the simulation technique was not crucial. However in the current work, we would crucially exploit the fact that the simulation strategy was straightline.

some such challenge is inherent in the light of the long list of general impossibility results known [Lin08, BPS06]. Before we describe the challenge faced in detail, it would be helpful to recall how the non-black box techniques based on [Bar01] work at a high level.

- **Non-black box technique.** In each session, the simulator has to commit to a program Π, which has to generate the adversary's random string r in that session. In the transcript between the commitment to Π and r, there may be messages of other sessions, which Π has to regenerate. Even if the program Π consists of the entire state of the simulator and the adversary at the point of the commitment, it runs into a problem in the case of secure computation (where the adversary is getting non-trivial output in each session).
- **Key challenge.** Note that to reach from the commitment of Π to the message r, the simulator makes use of some external information: namely the outputs it learns by querying the ideal functionality as it proceeds in the simulation. This information, however, is not available with the program Π (since the simulator may query the ideal functionality *after* the program Π was committed to). Also, note that the number of outputs learnt could be any unbounded polynomial. Hence, it is not clear how to regenerate the transcript.

The first obvious solution, which does not work, is to allow the program Π to take inputs of unbounded length. This would allow the simulator to pass all the outputs obtained to the program Π. But now the soundness of the protocol seems to be completely compromised. On the other hand, if Π does not receive all the outputs, it cannot regenerate the transcript!

To resolve this issue, we use the idea of "Oracle programs" due to Deng, Goyal, and Sahai [DGS09]. The program Π, while running, is allowed to make any (polynomially unbounded) number of queries (to be answered by the simulator) as long as the the response to each query is information theoretically fixed by the query. The soundness is still preserved: an adversarial prover still cannot communicate any information about the verifier's random string r to Π. However, the program Π can still access a potentially unbounded length string using such an "Oracle interface".

Unfortunately, *the above idea is still not sufficient for our purpose*: the outputs given by the ideal functionality are not fixed given the adversary's input in the session. Here we rely on the fact that we are only considering the ideal worlds which satisfy the bounded pseudoentropy condition. Very roughly, it is guaranteed that the entire output vector has only bounded pseudoentropy (B), given the input of the adversary. Moreover, given the adversary's input vector, all possible output vectors are efficiently testable by the PPT algorithm $\mathcal{T}$. In other words, for every vector of queries, there is only a bounded (although potentially *exponential*) number of response vectors accepted by $\mathcal{T}$. We allow the program Π to make any number of queries such that the response vector is accepted by $\mathcal{T}$. More details regarding our precise language for non-black box simulation may be found in Figure 1. This idea allows the simulator to supply the entire output vector (learnt from the ideal functionality) to Π while still preserving soundness. The soundness proof relies on the fact that the queries only allow for communication of up to B-bit string to Π, which is still not sufficient for communicating the string r.

Finally, there are additional challenges due to the requirement of straightline extraction. Towards that end, we rely on input indistinguishable computation introduced by Micali, Pass, and Rosen [MPR06]. Challenges also arise with performing hybrid arguments in the setting where the code of the simulator itself is committed (because of non-black box simulation). The full construction along with the main ideas is given in Appendix 4.

Other Related Work: Though Goyal et al. [Goy13] gave the first protocol for concurrent zero-knowledge with a straightline simulator, recently, Chung et al. [CLP13b] gave a *constant round* concurrent zero-knowledge protocol for uniform adversaries based on a new assumption of P-certificates, which is also straightline simulatable. Their protocol represents an exciting idea which opens an avenue for getting *constant round* concurrently secure computation protocols (albeit for uniform adversaries only, and, based on a new assumption). We believe that our techniques could also be applicable in constructing concurrent secure computation protocols using the protocol of [CLP13b].

2 Concurrently Secure Computation: Our Model

In this section, we begin by giving a brief sketch of our model. Formal description is given in Appendix B (building upon the model of [Lin08]). In this work, we consider a malicious, static and probabilistic polynomial time adversary that chooses whom to corrupt before the execution of the protocol and controls the scheduling of the concurrent executions. Additionally, the adversary can choose the inputs of different sessions adaptively. We denote the security parameter by n. We give a real world/ideal world based security definition. There are k parties $Q_1, Q_2, \ldots, Q_k$, where each party may be involved in multiple sessions with possibly interchangeable roles. In the ideal world, there is a trusted party for computing the desired two-party functionality $\mathcal{F} : \{0,1\}^{r_1} \times \{0,1\}^{r_2} \rightarrow \{0,1\}^{s_1} \times \{0,1\}^{s_2}$. Let the total number of executions be $m = m(n)$. Note that there is no a-priori bound on the number of sessions m and the adversary can start any (possibly unbounded) polynomial number of sessions. On the other hand, in the real world there is no trusted party and the two parties involved in a session, say P_1 and P_2, execute a two party protocol Π for computing $\mathcal{F}$. Our security definition (see Definition 3, Appendix B) requires that any adversary in the real model can be *emulated* by an adversary in the ideal model.

2.1 Our Result and its Applications.

As mentioned in the introduction, our main result (see Theorem 1, Section 1.1) is a general positive result for concurrent secure computation as long as the ideal world satisfies the bounded pseudo-entropy condition (Definition 1, Section 1.1).

Next, we show that our theorem not only subsumes the positive results of [Goy12] in the single input setting but also improves the round complexity.

Comparing our results with [Goy12]. In [Goy12], Goyal showed that if the ideal world satisfies the "key technical property" (KTP), then there exists a real world protocol which securely realizes this ideal world. The key technical property, taken verbatim from [Goy12], is as follows:

Definition 2 (Key technical Property (definition 3, [Goy12])) *The key technical property (KTP) of an ideal world experiment requires the existence of a PPT predictor $\mathcal{P}$ satisfying the following conditions. For all sufficiently large n, there exists a bound D such that for all adversaries and honest party inputs,*

$$\left|\left\{j \ : \ \mathcal{P}(\{I[\ell]\}_{\ell \leq j}, \{O[\ell]\}_{\ell < j}) \ \neq \ O[j]\right\}\right| < D$$

For the ideal worlds which satisfy KTP, [Goy12] gave a $O(n^3 D^2)$ round secure protocol which realizes the functionality, where D is the parameter in Definition 2.

We show that if an ideal world experiment satisfies the key technical property, then it also satisfies the bounded pseudoentropy condition.

Lemma 1 *If an ideal world experiment satisfies the key technical property (Definition 2), then it also satisfies the bounded pseudoentropy condition (Definition 1).*

For the proof of this lemma refer to Appendix C.

As mentioned before, the round complexity of Goyal [Goy12] is $O(n^3D^2)$ which is a polynomial in security parameter n as well as D (which depends upon length of single input as well as nature of functionality). Our Theorem 1 and Lemma 1 imply a quantitative and qualitative improvement in round complexity. This leads to lower round protocols for applications like private database search, secure set intersection, computing k^{th} ranked element etc. For details see Appendix D.

Moreover, [Goy12] only gave a positive result for functionalities with hardness free ideal world, i.e. in the ideal world the trusted party is not required to perform any cryptographic operations. There is no such restriction in our setting. In fact, we show that blind signatures and verifiable random functions satisfy the bounded pseudoentropy condition. More interestingly, they do not satisfy the key technical property. We next describe our results for these functionalities.

Blind Signatures. Blind signatures, introduced by [Cha82], allow users to obtain signatures on messages of their choice without revealing the message being signed to the signer (blindness property). In addition, they also need to satisfy the unforgeability property of the digital signature schemes. In this work, we give the following positive result for concurrent blind signatures.

Theorem 2 *Assume the existence of collision resistant hash functions and constant-round semi-honest oblivious transfer. Then for any constant ϵ, there exists a $O(n^\epsilon)$ round secure protocol which realizes the ideal world for concurrent blind signature functionality.*

We prove this theorem by using unique signatures [GO92] as the underlying signature scheme and showing that blind signatures satisfy the bounded pseudoentropy condition when the underlying signature scheme is unique. (Note that Lindell's black box impossibility result also holds in this setting.) A signature scheme is said to be *unique* if for each public key and each message, there exists at most one valid signature which verifies.

We can model blind signature as a two party computation between the signer and the user for the circuit for generating signatures. Note that the circuit will have the verification key vk hardcoded. At the end of the protocol, the user outputs a valid signature σ if obtained, and signer always outputs $\perp$. Now we show that this functionality satisfies BPC for $B = 0$ and $\mathcal{T}$ algorithm which is same as the signature verification algorithm. Note that if the adversary is playing the role of the user, its output is unique and is completely determined by its input message since vk is fixed by the function being computed. If the adversary is playing the role of the signer, its output is always $\perp$. Hence, set S will contain only one output vector, which is information theoretically fixed by the adversary inputs and the ideal world experiment (which fixes the verification keys for all the sessions). The algorithm $\mathcal{T}$ simply verifies the user's signatures w.r.t. corresponding vk and ensures that signer's outputs are $\perp$.

Finally note that blind signatures will not satisfy the key technical property. Consider the case when the adversary is acting as the user in all the sessions. By the unforgeability property of the scheme, any PPT predictor which receives k valid input/output (message/signature) pairs *cannot*

predict the signature on the next message with non-negligible probability. Also, note that blind signatures will not satisfy the generalized key technical property discussed in the full version [Goy11] for the same reason. For more formal description see Appendix D.1.

Verifiable Random Functions. Verifiable random functions (VRFs) were introduced by Micali, Rabin, and Vadhan [MRV99]. They combine the properties of pseudo-random functions with the verifiability property. Intuitively, they are pseudo-random functions with a public key and proofs for verification. Along with pseudo-randomness, they are required to satisfy *uniqueness*, i.e., given the public key, for any input x, there is a unique y which can verify. In this work, we show the following:

Theorem 3 *Assume the existence of collision resistant hash functions and constant-round semi-honest oblivious transfer. Then for any constant ϵ, there exists a $O(n^\epsilon)$ round concurrent real world protocol which realizes the ideal world experiment for verifiable random functions.*

We again prove this theorem by showing that VRFs satisfy BPC for $B = 0$ and $\mathcal{T}$ algorithm which is same as verification algorithm. Here, we again rely on the uniqueness property. Finally, note that VRFs too will not satisfy the key technical property due to pseudo-randomness guarantee. For details see Appendix D.2.

3 Our Simulation-Sound Non-Black Box Zero-knowledge Protocol

Constructing a family of polynomially many zero-knowledge protocols which are *simulation-sound* with respect to each other under (unbounded polynomially many) concurrent executions is one of the difficulties in constructing protocols for fully concurrent multi-party computation (MPC). Simulation-soundness, introduced by Sahai [Sah99], means that the soundness of each of the proofs given by the adversary should hold even when the adversary is getting unbounded polynomial number of simulated proofs. To avoid the problem of providing multiple outputs due to a rewinding based simulator for concurrent MPC, we need to construct simulation-sound zero-knowledge protocols which are straight-line simulatable. Note that Pass [Pas04] also gave a construction of polynomially many protocols which are concurrent zero-knowledge and simulation-sound w.r.t. each other in the restricted setting of bounded concurrency. In this work, we construct such simulation-sound zero-knowledge protocols building upon the non-black box public coin concurrent zero-knowledge protocol of Goyal [Goy13].

First, we give a brief overview of [Goy13]. Some of the text has been taken verbatim from [Goy13]. One of the main technical ideas in [Goy13] is to have $N = n^\epsilon$ non-black box slots, for any constant ϵ (each consisting of a commitment to a machine and a verifier challenge string). Each slot is followed by a universal argument (UA) execution. Any of the UA's in a session may be picked for simulation. If a UA is picked for simulation, to make the analysis go through, the simulator could choose of any of the previously completed slots and prefer the slots which are computationally lighter. In a UA execution, the prover proves that in one of the completed slots, the machine committed successfully outputs the verifier challenge string. Other main idea was to have encrypted executions of the UAs (using its public coin property) to hide the location of the convincing UA executions in the transcript. Finally there is an execution of a witness-indistinguishable argument of knowledge (WIAOK), where the prover proves that either the statement $x \in L$ or there exists a decryption of one of the UAs which is accepting. In the subsequent discussion, we will refer to the

The language Λ is defined w.r.t. an algorithm $\mathcal{T}$ and bound B with the following property: For any vector $\vec{x}$ (of possibly unbounded polynomial length) there exists a set S containing vectors $\vec{y}$ such that $|S| \leq 2^B$ and for all $\vec{y}' \notin S, \mathcal{T}(\vec{x}, \vec{y}') = 0$. Now the language Λ is defined as follows:
We say that $(h, z, r) \in \Lambda$ if there exists an oracle program Π s.t. $z = \text{COM}(h(\Pi))$ and there exist strings $y_1 \in \{0,1\}^{\leq |r| - B - n}$, $y_2 \in \{0,1\}^{\leq n^{\text{loglog}n}}$ and $y_3 \in \{0,1\}^{\leq n^{\text{loglog}n}}$ with the following properties. The oracle program Π takes y_1 as input and outputs r within $n^{\text{loglog}n}$ steps. Program Π can make two kinds of calls to the oracle

1. Produce a query of the form $\mathsf{decommit}(\mathsf{str})$ and expecting (r) with $\mathsf{str} = \text{COM}(r)$ in return such that the tuple (str, r) is guaranteed to be found in the string y_2 (as per a suitable encoding of y_2). Thus, such oracle calls by Π can be answered using y_2.

2. Produce a query of the form $\mathsf{output}(x)$ and expecting y in return, such that the tuple (x, y) is guaranteed to be found in the string y_3 (as per a suitable encoding of y_3). Thus, such oracle calls by Π can be answered using y_3.

If the program Π makes a query that cannot be answered by strings y_2 or y_3, Π aborts and we have that $(h, z, r) \notin \Lambda$. Also, let $\vec{x}$ denote the vector containing all the $\mathsf{output}(\cdot)$ queries made by Π (throughout its execution) and $\vec{y}$ be the corresponding responses, then Π aborts if $\mathcal{T}(\vec{x}, \vec{y}) = 0$ and we have that $(h, z, r) \notin \Lambda$.

Figure 1: Our language for zero-knowledge with non-black-box simulation

part of the protocol with non black box slots and encrypted UAs as the *preamble phase* and last phase as the WIAOK phase.

Two main ideas are required to transform the above described protocol into simulation-sound zero-knowledge protocols, which can then be used to construct protocols for concurrent MPC. Firstly, observe that unless the parties have identities it is impossible to construct a simulation-sound protocol because a man-in-the-middle attack cannot be prevented. Hence, we focus on a setting where each party has a unique identity of n bits. Let NMCom be a k-robust identity-based non-malleable commitment scheme (Appendix A.6). Now, after the preamble phase of the protocol, the prover with identity id gives a non-malleable commitment to the witness under its identity id. More precisely, the prover, having witness w to $x \in L$, gives a commitment $c = \mathsf{NMCom}(w)$ under his identity id. In the final WIAOK phase, the prover proves that either there exists a w such that $c = \mathsf{NMCom}(w)$ and $w \in R_L(x)$ or one of the UA executions was convincing. We will be able to prove the simulation-soundness of our protocol using the non-malleability and k-robustness of NMCom. Note that (as described later) our protocol will be simulation-sound even when the adversary is allowed to choose the statements to be proven adaptively till the point when he gives this non-malleable commitment.

Secondly, in our UA executions we will use a special generalized language Λ (see Figure 1) for the UA executions. Here, along with [DGS09] kind of queries $\mathsf{decommit}(\cdot)$ whose response is information theoretically fixed given the query itself, we will also have a second kind of queries, which we will denote by $\mathsf{output}(\cdot)$. Note that though the responses of these queries is not information theoretically fixed, they have a bounded pseudoentropy. Next, we give some intuition about the use of these oracle queries.

Intuition behind the oracle queries $\mathsf{output}(\cdot)$ in language Λ. The algorithm $\mathcal{T}$ and the bound B are introduced to capture the information learnt by the adversary. When only concurrent ses-

sions of zero-knowledge are running, there is no information passed to the adversary, hence we can have $\mathcal{T}$ to reject all outputs and still be able to simulate the view of the adversary. This notion will be important for the concurrent executions of multiparty computation because the adversary learns non-trivial information from calls to the trusted party. In particular, it learns the output of the function in each session. We will use the oracle queries $\mathsf{output}(\cdot)$ to communicate the information learnt from the trusted party to the adversary in the ideal world. But still to get our positive result, we will need to bound the amount of information learnt by the adversary. The bound B will be the number of bits of information passed on to the adversary. This is intuitively captured by the condition that there are only 2^B vectors of oracle responses which might be accepted by $\mathcal{T}$. Looking ahead, the description of $\mathcal{T}$ will depend on the functionality being computed.

Formal Protocol Description. Let $\text{COM}(\cdot)$ denote a non-interactive perfectly binding commitment scheme (Appendix A). Whenever we need to be explicit about the randomness, we denote by $\text{COM}(s; r)$ a commitment to a string s computed with randomness r. Unless stated otherwise, all commitments in the protocol are executed using this commitment scheme. Let NMCom be the k-robust non-malleable commitment scheme (Appendix A.6), where k is a parameter computed later. Let $\mathsf{len} = n^2 + B + \eta$, where B and η are parameters computed later.

The common input to P and V is the security parameter n. The input to P is x in the language $L \in NP$, and a witness w to $x \in L$. Let id be the n bit identity of the prover. Our protocol $\langle P, V \rangle$ or $c\mathcal{ZK}_{\mathsf{id}}$, where id is the identity of the prover, proceeds as follows: Parts of the protocol have been taken verbatim from [Goy13].

1. The verifier V chooses a random collision resistant hash function h from a function family $\mathcal{H}$ and sends it to P.
2. For $i \in [n^6]$, the protocol proceeds as follows:[4]
   - The prover P computes $z_i = \text{COM}(h(0))$ and sends it to V.
   - The verifier V selects a challenge string $r_i \xleftarrow{\$} \{0,1\}^{\mathsf{len}}$ and sends it to the prover P. The above two messages (consisting of the prover commitment and the verifier challenge) are referred to as a "slot".
   - The prover P and the verifier V will now start a three-round public coin universal argument (of knowledge) [BG02] where P proves to V that *there exists* $j \leq i$, *s.t.,* $\tau_j (= (h, z_j, r_j))$ *is in the language* Λ (see figure 1).

     The three messages of this UA protocol are called as the *first UA message*, *verifier UA challenge*, and, the *last UA message*.

     Observe that the UA does not just refer to the slot immediately preceding it but rather has a choice of using *any of the slots that have completed* in the protocol so far.
   - The prover computes the first UA message and sends a *commitment* to this message to the verifier. The honest prover will simply commit to a random string of appropriate size.
   - The verifier now sends the UA challenge message.
   - The prover computes the last UA message and again sends only a *commitment* to this message to the verifier. The honest prover will simply commit to a random string of

[4]Note that the round complexity of our protocol can be made n^ϵ using standard techniques involving "scaling down" the security parameter.

appropriate size.

3. The prover declares the statement $x \in L$ and commits to the witness w using the non-malleable commitment scheme NMCom under prover's identity id.

   Note that a cheating prover can adaptively choose the statement x here.

4. Finally, the prover proves the following statement to V using WIAOK

   1. The value committed to in Step 3 is a value w such that it is a valid witness to $x \in L$, (i.e. $w \in R_L(x)$), *or*
   2. There exists i such that the i-th UA execution was "convincing". That is, there exists an $i \in [n^6]$ such that there exists an opening to the prover first and last UA messages such that an honest verifier would have accepted the transcript of the UA execution.

   An honest prover simply commits to the witness for $x \in L$ in Step 3 and uses the first part of the statement to complete the witness-indistinguishable argument of knowledge protocol.

   Observe that a witness to the second part of the above statement would be the opening of the commitments to the UA first and last messages. Hence, the size of the witness is fixed and depends only upon the communication complexity of the 3-round UA system being used.

Remark 1 *We call the Steps 1 and 2 of the protocol as non-black box* preamble*, step 3 as the* NMCOM *phase and step 4 as the* WIAOK *phase.*

Parameter k. We set k to be the round complexity of WIAOK. Hence, we set $k = 3$.

Parameter B. Note that the parameter B in len is same as the one in Figure 1, i.e. the parameter specified for algorithm $\mathcal{T}$ in the description of language Λ.

Setting the parameter η. Let η be the sum of the following: prover's maximum communication complexity in different primitives used in the protocol described above, and communication complexity of NMCom. More precisely, we set

$$\eta = \mathsf{max}(c_z, c_{\text{UA1}}, c_{\text{UA2}}, c_{\text{WIAOK}}, c_{\mathsf{NMCom},S}) + c_{\mathsf{NMCom},R},$$

where c_z is the length of the the slot begin message z, c_{UA1} is the length of the UA first message, c_{UA2} is the length of the UA last message, c_{WIAOK} is the prover's communication complexity in the final WIAOK execution, $c_{\mathsf{NMCom},S}$ is the sender's communication complexity in NMCom and $c_{\mathsf{NMCom},R}$ is the receiver's communication complexity in NMCom.

Looking ahead, (very informally) while proving the simulation-soundness of the above protocol, different parts of the protocol will be taken externally and NMCom given by the adversary will be exposed to an external receiver, etc. Hence, different parts of the protocol will be given externally to the machine committed by the simulator as part of the string y_1 in Λ.

Note that the entire $\langle P, V \rangle$ protocol is run w.r.t. to language Λ having a specific algorithm $\mathcal{T}$ and bound B. We will prove that the security properties hold for any such $\mathcal{T}$ and bound B when η is chosen as above. Next, we prove the soundness of the protocol for any fixed value of B. Then we will prove the simulation-soundness of the protocol. Our ZK simulator will not use the oracle queries of the type $\mathsf{output}(\cdot)$. Later on our MPC simulator will make a non-trivial use of these oracle queries.

The proof of security of simulation-sound non-black box zero-knowledge protocol proceeds along the lines discussed in the introduction (see Section 1.2). We give a detailed formal proof of security in Appendix E.

4 Concurrently Secure Computation: Our Protocol

In this section, we will describe our protocol Σ for concurrently secure computation for ideal world experiments which satisfy the bounded pseudoentropy condition (Definition 1) for some parameter $B \in \mathbb{N}$ and algorithm $\mathcal{T}$.

Our Construction. In order to describe our construction, we first recall the notation associated with the primitives that we use in our protocol. Let $\text{COM}(\cdot)$ denote the commitment function of a non-interactive perfectly binding commitment scheme (Appendix A). Let $\langle P, V \rangle$ denote the simulation-sound non-black box concurrent zero-knowledge protocol as described in Section 3 with length of challenge strings modified to be $\mathsf{len} = n^2 + B + \theta$, where θ is a parameter computed later. Let $\langle P_1^{\mathsf{iic}}, P_2^{\mathsf{iic}} \rangle$ be the constant round protocol for input indistinguishable computation described in Appendix A.5. Let NMCom be the k-robust non-malleable commitment scheme (Appendix A.6), where k is a parameter computed later. Further, let $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle$ denote a witness indistinguishable argument (Appendix A.3) and let $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ denote a constant round *semi-honest* two party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ that securely computes $\mathcal{F}$ in the stand-alone setting as per the standard definition of secure computation (Appendix A.4).

Let P_1 and P_2 be two parties with inputs x_1 and x_2. Let n be the security parameter. Protocol $\Sigma = \langle P_1, P_2 \rangle$ proceeds as follows:

I. Non-Black Box Simulation Phase.

1. $P_1 \Rightarrow P_2$: P_1 and P_2 engage in the *preamble* phase of $\langle P, V \rangle$ where P_1 is the prover. Next, in the NMCOM phase, P_1 creates a non-malleable commitment com_1 to bit 0, i.e. $\mathsf{com}_1 = \mathsf{NMCom}(0)$ and sends com_1 to P_2. P_1 and P_2 now engage in the WIAOK phase where P_1 proves that either (1) com_1 is a commitment to 0 , or (2) there exists i such that the i-th UA execution in the preamble phase was "convincing".

2. $P_2 \Rightarrow P_1$: P_2 now acts symmetrically. P_1 and P_2 engage in the *preamble* phase of $\langle P, V \rangle$ where P_2 is the prover. Next, P_2 creates a non-malleable commitment com_2 to bit 0, i.e. $\mathsf{com}_2 = \mathsf{NMCom}(0)$ to bit 0 and sends com_2 to P_1. P_1 and P_2 now engage in the WIAOK phase where P_2 proves that either (1) com_2 is a commitment to 0 , or (2) there exists i such that the i-th UA execution in the preamble phase was "convincing".

Informally speaking, the purpose of this phase is to aid the simulator in obtaining a "trapdoor" to be used during the simulation of the other two phases of the protocol.

Common input: Let $\text{COM}(\cdot)$ be a non-interactive perfectly binding commitment scheme. The functionality $f_{\mathsf{com}_1,\mathsf{com}_2}$ is parameterized by two commitments com_1 and com_2 under $\text{COM}(\cdot)$, which are the common inputs to the functionality and the parties P_1^{iic} and P_2^{iic}.
Inputs: Let (z_1, td_1) and (z_2, td_2) be the inputs of P_1^{iic} and P_2^{iic} respectively.

Computation: Party P_1^{iic} sends its input (z_1, td_1) and party P_2^{iic} sends its input (z_2, td_2) to the trusted functionality $f_{\mathsf{com}_1,\mathsf{com}_2}$.
If td_1 is a *valid* opening of com_1 to bit 1, $f_{\mathsf{com}_1,\mathsf{com}_2}$ sends z_2 to P_1^{iic}, otherwise it sends $\perp$. Similarly, if td_2 is a *valid* opening of com_2 to bit 1, $f_{\mathsf{com}_1,\mathsf{com}_2}$ sends z_1 to P_2^{iic}, otherwise it sends $\perp$.

Figure 2: The Functionality $f_{\mathsf{com}_1,\mathsf{com}_2}$

II. Input Indistinguishable Computation Phase. Intuitively speaking, in this phase, the parties "commit" to their inputs and random coins (to be used in the final secure computation phase) by engaging in a execution of $\langle P_1^{\mathsf{iic}}, P_2^{\mathsf{iic}} \rangle$ for the functionality $f_{\mathsf{com}_1,\mathsf{com}_2}$ described in Figure 2. More precisely, P_1 and P_2 engage in an execution of $\langle P_1^{\mathsf{iic}}, P_2^{\mathsf{iic}} \rangle$ for the functionality $f_{\mathsf{com}_1,\mathsf{com}_2}$ where P_1 plays the role of P_1^{iic}, while P_2 plays the role of P_2^{iic} as follows:

1. P_1 first samples a random string r_1 (of appropriate length, to be used as P_1's randomness in the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ in Phase III) and uses input $z_1 = x_1 \| r_1$ and $\mathsf{td}_1 = \perp$ in execution of $\langle P_1^{\mathsf{iic}}, P_2^{\mathsf{iic}} \rangle$ for $f_{\mathsf{com}_1,\mathsf{com}_2}$.

2. $P_2 \Rightarrow P_1 : P_2$ now acts symmetrically. P_2 first samples a random string r_2 (of appropriate length, to be used as P_2's randomness in the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ in Phase III) and uses input $z_2 = x_2 \| r_2$ and $\mathsf{td}_2 = \perp$ in execution of $\langle P_1^{\mathsf{iic}}, P_2^{\mathsf{iic}} \rangle$ for $f_{\mathsf{com}_1,\mathsf{com}_2}$.

Informally speaking, the purpose of this phase is to aid the simulator in extracting the adversary's input and randomness with the help of the trapdoor obtained in the previous phase. As we will show later, an adversary will never be able to input a valid trapdoor.

III. Final Secure Computation Phase.[5] In this phase, P_1 and P_2 engage in an execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ where P_1 plays the role of P_1^{sh}, while P_2 plays the role of P_2^{sh}. Since $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ is secure only against semi-honest adversaries, parties first run a coin-flipping protocol to enforce that the coins of each party are truly random. We then compile the semi-honest $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ with $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle$ to ensure correct behavior on part of each party. More precisely, after sending each protocol message, a party also gives a proof using $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle$ that the message generated is consistent with the transcript so far and the input used in the previous phase. More precisely, this phase proceeds as follows:

1. $P_1 \leftrightarrow P_2 : P_1$ samples a random string r_2' (of same length as r_2) and sends it to P_2. Similarly, P_2 samples a random string r_1' (of same length as r_1) and sends it to P_1. Let $r_1'' = r_1 \oplus r_1'$ and $r_2'' = r_2 \oplus r_2'$. Now, r_1'' and r_2'' are the random coins that P_1 and P_2 will use during the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$.

2. Let q be the number of rounds in $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$, where one round consists of a message from P_1^{sh} followed by a reply from P_2^{sh}. Let transcript $T_{1,j}$ (resp., $T_{2,j}$) be defined to contain all the messages exchanged between P_1^{sh} and P_2^{sh} before the point P_1^{sh} (resp., P_2^{sh}) is supposed to send a message in round j. For $j = 1, \ldots, q$:

   (a) $P_1 \Rightarrow P_2$: Compute $\beta_{1,j} = P_1^{\mathsf{sh}}(T_{1,j}, x_1, r_1'')$ and send it to P_2. P_1 and P_2 now engage in an execution of $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle$, where P_1 proves the following statement:

      i. *either* there exist values $\hat{x}_1$, $\hat{r}_1$ and $\hat{\mathsf{td}}_1$ such that (a) the $f_{\mathsf{com}_1,\mathsf{com}_2}$ is *valid* with respect to the value $\hat{z}_1 = \hat{x}_1 \| \hat{r}_1$ and $\hat{\mathsf{td}}_1$ and (b) $\beta_{1,j} = P_1^{\mathsf{sh}}(T_{1,j}, \hat{x}_1, \hat{r}_1 \oplus r_1')$

      ii. *or*, the non-malleable commitment com_1 is a commitment to bit 1.

   (b) $P_2 \Rightarrow P_1 : P_2$ now acts symmetrically.

This completes the description of the protocol $\Sigma = \langle P_1, P_2 \rangle$. Note that Π consists of several instances of WI, such that the proof statement for each WI instance consists of two parts. Specifically,

[5] Part of the text in this phase has been taken verbarim from [GGJS12]

the second part of the statement states that prover committed to bit 1 in the non-black box simulation phase. In the sequel, we will refer to the second part of the proof statement as the *trapdoor* condition. Further, we will call the witness corresponding to the first part of the statement as *real* witness and that corresponding to the second part of the statement as the *trapdoor* witness.

Parameter k. We will set k to be the maximum round complexity among UA, WIAOK, $\langle P_1^{\mathsf{iic}}, P_2^{\mathsf{iic}} \rangle$ and $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$.

Setting the parameter θ. We will set θ to be the sum of the following: a party's maximum communication complexity in different primitives used in the protocol described above (excluding when it acts as a verifier in $\langle P, V \rangle$), and communication complexity of NMCom. More precisely, we set

$$\theta = \mathsf{max}(c_z, c_{\text{UA1}}, c_{\text{UA2}}, c_{\text{WIAOK}}, c_{\text{WI}}, c_{\text{IIC}}, c_{\text{TPC}}, c_{\mathsf{NMCom},S}) + c_{\mathsf{NMCom},R},$$

where c_z is the length of the message z (the slot begin message), c_{UA1} is the length of the UA first message, c_{UA2} is the length of the UA last message, c_{WIAOK} is the prover's communication complexity in the final WIAOK execution, c_{WI} is the prover's communication complexity in WI, c_{IIC} is the communication complexity of any party in $\langle P_1^{\mathsf{iic}}, P_2^{\mathsf{iic}} \rangle$, c_{TPC} is the total communication complexity of the semi-honest two party computation $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ for the functionality $\mathcal{F}$, $c_{\mathsf{NMCom},S}$ is the sender's communication complexity in NMCom and $c_{\mathsf{NMCom},R}$ is the receiver's communication complexity in NMCom. Looking ahead, while proving the security of the above protocol, different parts of the protocol will be taken externally and NMCom given by the adversary will be exposed to external receiver, etc. Hence, all of these will be given externally to the machine committed by the simulator as part of the string y_1 in Λ.

The proof of Theorem 1 proceeds along the lines discussed in the introduction (see Section 1.2). We give a detailed formal description of the simulator and proof of security in Appendix F.

References

[AGJ+12] Shweta Agrawal, Vipul Goyal, Abhishek Jain, Manoj Prabhakaran, and Amit Sahai. New impossibility results for concurrent composition and a non-interactive completeness theorem for secure computation. In *CRYPTO*, pages 443–460, 2012.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS*, pages 186–195, 2004.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *IEEE Conference on Computational Complexity*, pages 194–203, 2002.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. *Proceedings of the International Congress of Mathematicians*, 1986.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography*, pages 31–46, 2003.

[BPS06] Boaz Barak, Manoj Prabhakaran, and Amit Sahai. Concurrent non-malleable zero knowledge. In *FOCS*, pages 345–354, 2006.

[BS05] Boaz Barak and Amit Sahai. How to play almost any mental game over the net - concurrent composition via super-polynomial simulation. In *FOCS*, pages 543–552, 2005.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[CKL03] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In *EUROCRYPT*, pages 68–86, 2003.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires $\widetilde{\Omega}(\log n)$ rounds. In *STOC*, pages 570–579, 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.

[CLP10] Ran Canetti, Huijia Lin, and Rafael Pass. Adaptive hardness and composable security from standard assumptions. In *FOCS*, 2010.

[CLP13a] Ran Canetti, Huijia Lin, and Omer Paneth. Public-coin concurrent zero-knowledge in the global hash model. In *TCC*, pages 80–99, 2013.

[CLP13b] Kai-Min Chung, Huijia Lin, and Rafael Pass. Constant-round concurrent zero knowledge from p-certificates. In *FOCS*, pages 50–59, 2013.

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

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *STOC*, pages 409–418, 1998.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In *CRYPTO*, pages 60–77, 2006.

[GG14] Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In *Eurocrypt*, pages 477–495, 2014.

[GGJ13] Vipul Goyal, Divya Gupta, and Abhishek Jain. What information is leaked under concurrent composition? In *CRYPTO (2)*, pages 220–238, 2013.

[GGJS11] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. *IACR Cryptology ePrint Archive*, 2011:251, 2011.

[GGJS12] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Concurrently secure computation in constant rounds. In *EUROCRYPT*, pages 99–116, 2012.

[GJ13] Vipul Goyal and Abhishek Jain. On concurrently secure computation in the multiple ideal query model. In *EUROCRYPT*, pages 684–701, 2013.

[GJO10] Vipul Goyal, Abhishek Jain, and Rafail Ostrovsky. Password-authenticated session-key generation on the internet in the plain model. In *CRYPTO*, pages 277–294, 2010.

[GKOV12] Sanjam Garg, Abishek Kumarasubramanian, Rafail Ostrovsky, and Ivan Visconti. Impossibility results for static input secure computation. In *CRYPTO*, pages 424–442, 2012.

[GL01] Oded Goldreich and Yehuda Lindell. Session-key generation using human passwords only. In *CRYPTO*, pages 408–432, 2001.

[GM00] Juan A. Garay and Philip D. MacKenzie. Concurrent oblivious transfer. In *FOCS*, pages 314–324, 2000.

[GM11] Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In *FOCS*, pages 678–687, 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 *STOC*, pages 218–229, 1987.

[GO92] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In *CRYPTO*, pages 228–245, 1992.

[Goy11] Vipul Goyal. Positive results for concurrently secure computation in the plain model. *IACR Cryptology ePrint Archive*, 2011:602, 2011.

[Goy12] Vipul Goyal. Positive results for concurrently secure computation in the plain model. In *FOCS*, pages 41–50, 2012.

[Goy13] Vipul Goyal. Non-black-box simulation in the fully concurrent setting. In *STOC*, pages 221–230, 2013.

[GRS+11] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In *CRYPTO*, pages 630–648, 2011.

[GS09] Vipul Goyal and Amit Sahai. Resettably secure computation. In *EUROCRYPT*, pages 54–71, 2009.

[HKKL07] Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In *TCC*, pages 323–341, 2007.

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT*, pages 115–128, 2007.

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

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

[KZ06] Aggelos Kiayias and Hong-Sheng Zhou. Concurrent blind signatures without random oracles. In *SCN*, pages 49–62, 2006.

[Lin03a] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *STOC*, pages 683–692, 2003.

[Lin03b] Yehuda Lindell. General composition and universal composability in secure multi-party computation. In *FOCS*, pages 394–403, 2003.

[Lin08] Yehuda Lindell. Lower bounds and impossibility results for concurrent self composition. *J. Cryptology*, 21(2):200–249, 2008.

[Lin11] Huijia Rachel Lin. Concurrent security. Cornell Ecommons Library, 2011. `http://ecommons.library.cornell.edu/bitstream/1813/29141/1/hl359thesisPDF.pdf/`.

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

[Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the dh-ddh separation. In *CRYPTO*, pages 597–612, 2002.

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *FOCS*, pages 367–378, 2006.

[MR01] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In *CRYPTO*, pages 542–565, 2001.

[MR02] Silvio Micali and Ronald L. Rivest. Micropayments revisited. In *CT-RSA*, pages 149–163, 2002.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.

[Oka06] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In *TCC*, pages 80–99, 2006.

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

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *STOC*, pages 232–241, 2004.

[PPS+08] Omkant Pandey, Rafael Pass, Amit Sahai, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Precise concurrent zero knowledge. In *EUROCRYPT*, pages 397–414, 2008.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *FOCS*, pages 404–413, 2003.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251, 2004.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

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

A Preliminaries

We now briefly mention some of the main cryptographic primitives that we use in our construction. The first two definitions have been taken verbatim from [Goy13].

A.1 Non-interactive perfectly binding commitment scheme with a unique decommitment

In our protocol, we shall use a non-interactive perfectly binding commitment scheme with the properties that every commitment has a unique decommitment and the verification of the decommitment is deterministic. An example of such a scheme is the scheme that commits to the bit b by $\textsc{com}(b;(r,x)) = r||\pi(x)||(x \cdot r) \oplus b$ where π is a one-to-one one-way function on the domain $\{0,1\}^n$, $x \cdot y$ denotes the inner-product of x and y over $GF(2)$, and $x, r \leftarrow U_n$. We denote this commitment scheme by $\textsc{com}$.

As pointed out in [Goy13], we can remove the 1-to-1 one-way function assumption in our work by relying on a simple trick from [CLP13a]. In particular, [CLP13a] uses a notion of forward-secure pseudo-random functions (based only on one-way functions) which can also be employed in our construction (as opposed to non-interactive commitment schemes with a unique decommitment).

A.2 Three-round public-coin universal-arguments

Universal arguments [BG02] are used in order to provide efficient proofs to statements of the form $y = (M, x, t)$, where y is considered to be a true statement if M is a non-deterministic machine that accepts x within t steps. We shall make use of public-coin universal-arguments from [BG02] which is only 3-rounds assuming the prover and the verifier have agreed upon a function from a CRHF family before the protocol starts. In addition, we shall make use of the fact that the universal argument system from [BG02] is a weak proof of knowledge.

Security notions. For definition of universal arguments and zero-knowledge argument of knowledge, we refer the reader to previous works [Bar01, BG02].

A.3 Witness Indistinguishable Arguments

In our construction, we shall use a witness indistinguishable (WI) argument $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle$ for proving membership in any **NP** language with perfect completeness and negligible soundness error. Three round WI arguments can be constructed by running $\omega(\log n)$ parallel copies of Blum's Hamiltonicity protocol [Blu86]. Moreover, it is a witness indistinguishable argument of knowledge.

A.4 Constant-Round Semi-Honest Two Party Computation

We will use a semi-honest two party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ that emulates the functionality $\mathcal{F}$ (as described in section B) in the stand-alone setting. The existence of such a protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ follows from the existence of constant-round semi-honest 1-out-of-2 oblivious transfer [Yao86, GMW87, Kil88].

A.5 Constant-Round Input Indistinguishable Computation (IIC)

Micali et al. [MPR06] introduced the notion of input indistinguishable computation (IIC) similar to witness indistinguishability for proofs. In IIC, very roughly, given the output vector (consisting of outputs in all concurrent sessions), consider any two honest party input vectors $\vec{x_1}$ and $\vec{x_2}$, consistent with the output vector. The security guarantee requires that the adversary only has a negligible advantage in distinguishing which of these is the actual honest party input vector. In IIC, no guarantees are provided for any two input vectors which don't lead to the identical output.

In this work, we will use the constant round IIC protocol by Garg et al. [GGJS12] which is secure even when unbounded number of executions are composed together under the assumption of constant round semi-honest oblivious transfer and collision resistant hash functions.

A.6 Non-Malleable Commitments Robust w.r.t. k-round Protocols

The notion of non-malleability w.r.t. arbitrary k-round protocols was introduced in [LP09]. Following text has been taken verbatim from [Lin11]. A non-malleable commitment scheme $\langle C, R \rangle$ is said to be robust w.r.t. to arbitrary k round protocols if the following holds: Consider a one-one man-in-the-middle adversary A that participates in one left interaction- communicating with a machine B- and one right interaction- acting as a committer using the commitment scheme $\langle C, R \rangle$. We denote by $\mathsf{nmc}^{B,A}_{\langle C,R \rangle}(y, z)$ the random variable consisting of the view of $A(z)$ in a man-in-the-middle execution when communicating with $B(y)$ on the left and an honest receiver on the right, combined with the value $A(z)$ commits to on the right. Intuitively, we say that $\langle C, R \rangle$ is non-malleable w.r.t. B if $\mathsf{nmc}^{B,A}_{\langle C,R \rangle}(y_1, z)$ and $\mathsf{nmc}^{B,A}_{\langle C,R \rangle}(y_2, z)$ are computationally indistinguishable whenever $B(y_1)$ and $B(y_2)$ are computationally indistinguishable. In our work, we will use commitment schemes which are non-malleable w.r.t. itself and arbitrary k-round protocols where k is a constant. Under the assumption that one way functions exist, for any constant k, there exists a constant-round commitment scheme that is k-robust concurrently non-malleable with a black box proof of security.

B Concurrently Secure Computation: Our Model

In this work, we consider a malicious, static adversary that chooses whom to corrupt before the execution of the protocol. The adversary controls the scheduling of the concurrent executions. We

only consider *computational* security and therefore restrict our attention to adversaries running in probabilistic polynomial time. We denote computational indistinguishability by $\stackrel{c}{\equiv}$, and the security parameter by n. We do not require fairness and hence in the ideal model, we allow a corrupt party to receive its output in a session and then optionally block the output from being delivered to the honest party, in that session. Further, we only consider "security with abort". To formalize the above requirements and define security, we follow the standard paradigm for defining secure computation (see also [Lin08]). We define an ideal model of computation and a real model of computation, and require that any adversary in the real model can be *emulated* by an adversary in the ideal model. More details follow.

IDEAL MODEL. We first define the ideal world experiment, where there is a trusted party for computing the desired two-party functionality $\mathcal{F} : \{0,1\}^{r_1} \times \{0,1\}^{r_2} \to \{0,1\}^{s_1} \times \{0,1\}^{s_2}$. Let P_1 and P_2 denote the two parties in a single execution. In total, let there be k parties $Q_1, Q_2, \ldots, Q_k$, where each party may be involved in multiple sessions with possibly interchangeable roles, i.e. Q_i may play the role of P_1 in one session and P_2 in some other session. Let the total number of executions be $m = m(n)$. For each $\ell \in [m]$, we will denote by P_1^ℓ, the party playing the role of P_1 in session ℓ. P_2^ℓ is defined analogously. The adversary may corrupt any subset of the parties in $Q_1, Q_2, \ldots, Q_k$. The ideal world execution proceeds as follows:

I. Inputs: There is a PPT *usage scenario* which gives inputs to all the parties. For each session $\ell \in [m]$, it gives inputs $x_\ell \in X \subseteq \{0,1\}^{r_1}$ to P_1^ℓ and $y_\ell \in Y \subseteq \{0,1\}^{r_2}$ to P_2^ℓ. The adversary is given auxiliary input $z \in \{0,1\}^*$, and chooses the subset of the parties to corrupt, say M. The adversary receives the inputs of the corrupted parties.

II. Session initiation: When the adversary wishes to initiate the session number ℓ, it sends a $(\mathsf{start\text{-}session}, \ell)$ message to the trusted party. On receiving a message of the form $(\mathsf{start\text{-}session}, \ell)$, the trusted party sends $(\mathsf{new\text{-}session}, \ell)$ to both P_1^ℓ and P_2^ℓ.

III. Honest parties send inputs to the trusted party: Upon receiving $(\mathsf{start\text{-}session}, \ell)$ from the trusted party, an honest party P_i^ℓ sends its real input along with the session identifier. More specifically, if P_1^ℓ is honest, it sends (ℓ, x_ℓ) to the trusted party. Similarly, an honest P_2^ℓ sends (ℓ, y_ℓ) to the trusted party.

IV. Corrupted parties send inputs to the trusted party: At any point during execution, a corrupted party P_1^ℓ may send a message (ℓ, x'_ℓ) to the trusted party, for any string x'_ℓ (of appropriate length) of its choice. Similarly, a corrupted party P_2^ℓ sends a message (ℓ, y'_ℓ) to the trusted party, for any string y'_ℓ (of appropriate length) of its choice.

V. Trusted party sends results to the adversary: For a session ℓ, when the trusted party has received messages from both P_1^ℓ and P_2^ℓ, it computes the output for that session. Let x'_ℓ and y'_ℓ be the inputs received from P_1^ℓ and P_2^ℓ, respectively. It computes the output $\mathcal{F}(x'_\ell, y'_\ell)$. If either P_1^ℓ or P_2^ℓ is corrupted, it sends $(\ell, \mathcal{F}(x'_\ell, y'_\ell))$ to the adversary. If neither of the parties is corrupted, then the trusted party sends the output message $(\ell, \mathcal{F}(x'_\ell, y'_\ell))$ to both P_1^ℓ and P_2^ℓ.

VI. Adversary instructs the trusted party to answer honest players: For a session ℓ, where exactly one of the party is corrupted, the adversary, depending on its view up to this point, may send the message (output, ℓ) to the trusted party. Then, the trusted party sends the output $(\ell, \mathcal{F}(x'_\ell, y'_\ell))$, computed in the previous step, to the honest party in session ℓ.

VII. Outputs: An honest party always outputs the value that it received from the trusted party. The adversary outputs an arbitrary (PPT computable) function of its entire view (including the view of all corrupted parties) throughout the execution of the protocol.

The ideal execution of a function $\mathcal{F}$ with security parameter n, input vectors $\vec{x}$, $\vec{y}$, auxiliary input z to $\mathcal{S}$ and the set of corrupted parties M, denoted by $\text{IDEAL}^{\mathcal{F}}_{M,\mathcal{S}}(n, \vec{x}, \vec{y}, z)$, is defined as the output pair of the honest parties and the ideal world adversary $\mathcal{S}$ from the above ideal execution.

REAL MODEL. We now consider the real model in which a real two-party protocol is executed (and there exists no trusted third party). Let $\mathcal{F}, \vec{x}, \vec{y}, z$ be as above and let Π be a two-party protocol for computing $\mathcal{F}$. Let $\mathcal{A}$ denote a non-uniform probabilistic polynomial-time adversary that controls any subset M of parties $Q_1, Q_2, \ldots, Q_k$. The parties run concurrent executions of the protocol Π, where the honest parties follow the instructions of Π in all executions. The honest party initiates a new session ℓ, using the input provided, whenever it receives a start-session message from $\mathcal{A}$. The scheduling of all messages throughout the execution is controlled by the adversary. That is, the execution proceeds as follows: the adversary sends a message of the form (ℓ, msg) to the honest party. The honest party then adds msg to its view of session ℓ and replies according to the instructions of Π and this view in that session. At the conclusion of the protocol, an honest party computes its output as prescribed by the protocol. Without loss of generality, we assume the adversary outputs exactly its entire view in the execution of the protocol.

The real concurrent execution of Π with security parameter n, input vectors $\vec{x}$, $\vec{y}$, auxiliary input z to $\mathcal{A}$ and set of corrupted parties M, denoted $\text{REAL}^{\Pi}_{M,\mathcal{A}}(n, \vec{x}, \vec{y}, z)$, is defined as the output pair of the honest parties and real world adversary $\mathcal{A}$, resulting from the above real-world process.

Definition 3 *Let $\mathcal{F}$ and Π be as above. Then protocol Π for computing $\mathcal{F}$ is a concurrently secure computation protocol if for every probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every polynomial $m = m(n)$, every input vectors $\vec{x} \in X^m$, $\vec{y} \in Y^m$, every $z \in \{0,1\}^*$, and every subset of corrupt parties M,*

$$\left\{\text{IDEAL}^{\mathcal{F}}_{M,\mathcal{S}}(n, \vec{x}, \vec{y}, z)\right\}_{n \in N} \stackrel{c}{\equiv} \left\{\text{REAL}^{\Pi}_{M,\mathcal{A}}(n, \vec{x}, \vec{y}, z)\right\}_{n \in N}$$

C Proof of Lemma 1

Proof. Since the ideal world experiment satisfies the key technical property, there exists a predictor $\mathcal{P}$ such that the number of adversary inputs for which it fails to predict the correct output (given all previous inputs/outputs) is at most $D - 1$. To show that this ideal world experiment also satisfies the bounded pseudoentropy condition, we need a bound $B \in \mathbb{N}$ and need to describe a PPT algorithm $\mathcal{T}$ and a set S which together satisfy the conditions in definition 1.

Existence of a predictor satisfying KTP implies that given an advice of at most $D - 1$ input/output pairs, it can predict all other outputs correctly. Using this intuition, we will build the set S as follows: Consider all possible strings str of the form $((i_1, O[i_1]), \ldots, (i_{D-1}, O[i_{D-1}]))$ such that $i_1, \ldots, i_{D-1}$ are sessions numbers in [m] and $O(i_j)$s are the corresponding possible outputs. Note that length of str, i.e., $|\mathsf{str}| \leq (D-1) \cdot (\log^2 n +$ size of output in any session). Let $|\mathsf{str}| = B$. Now we will construct the set S of size 2^B as follows. For each str as described above, we will construct a possible output vector $\vec{O}$ in S as follows. To compute the output for session number j check if index j is in str. If yes, it output for session number j is the corresponding value in str.

Else, run the predictor on the inputs and outputs computed so far and the input of the adversary in the current session, i.e. $O[j] \leftarrow \mathcal{P}(\{I[\ell]\}_{\ell \leq j}, \{O[\ell]\}_{\ell < j})$. Since there are 2^B possible advice strings str, the number of distinct output vectors in S is at most 2^B.

Next, we will describe the procedure $\mathcal{T}$. Given an input vector $\vec{I}$ and output vector $\vec{O}$, it does the following: It internally runs the predictor $\mathcal{P}$ session by session. For the session number j, it computes $\mathcal{P}(\{I[\ell]\}_{\ell \leq j}, \{O[\ell]\}_{\ell < j})$ and checks whether it is equal to $O[j]$ or not. Procedure $\mathcal{T}$ accepts the output vector $\vec{O}$ for input $\vec{I}$ if the predicted output does not match the corresponding value in $\vec{O}$ for at most $D-1$ locations. More precisely, $\mathcal{T}$ returns 1 if $\left|\left\{j \ : \ \mathcal{P}(\{I[\ell]\}_{\ell \leq j}, \{O[\ell]\}_{\ell < j}) \ \neq O[j]\right\}\right| < D$, else it returns 0.

Finally, we show that the set S and algorithm $\mathcal{T}$ described above satisfy the properties listed in definition 1. The first condition on the size of the set S is trivially satisfied.

For the second condition, note that since the predictor fails in predicting at most $D-1$ outputs of the adversary, and the set S contains all such output vectors which need advice for at most $D-1$ outputs, S does contain all the possible valid outputs for all possible inputs of the honest parties.

For the last condition, observe that the way the algorithm $\mathcal{T}$ would accept all the output vectors in the set S. It remain to prove that $\mathcal{T}$ rejects all the output vectors which are not in S. For this, it is sufficient to prove that for such vectors, the predictor $\mathcal{P}$ would fail on more than $D-1$ sessions. If $\mathcal{P}$ fails on less than D sessions, then there exists a advice string str using which all outputs could have been predicted. But then this output vector would have been in set S. Hence, proved.

■

D Applications satisfying Bounded Pseudoentropy Condition

We begin by listing the applications in [Goy12] which satisfy the key technical property for different values of D (Definition 2). We too give positive results for these by Lemma 1. Parts of the text below has been taken verbatim from [Goy12]. Then we describe the applications of blind signatures and verifiable random functions which satisfy the bounded pseudoentropy condition (BPC) but do not satisfy the key technical property (KTP). Hence, these are some of the applications for which we obtain the first positive result for concurrent composition.

Private Database search. In this scenario, the first (honest) party holds a database consisting of k entries. The second (adversarial) party has a predicate $g(\cdot)$ as input and gets as output all database entries on which this predicate evaluates to 1. [Goy12] show that the ideal world satisfies the KTP for $D = k + 1$. If each entry of the database has γ bits, by proof of lemma 1, the ideal world satisfies BPC for $B = k \cdot (\log^2 n + k\gamma)$.

To give more intuition about BPC, below we show independently that the functionality described satisfies BPC for $B = k\gamma$. Note that the total number of valid honest party inputs are $2^{k\gamma}$. Given the adversary's input vector, for each possible honest party input i.e. for each possible database with k entries, there is exactly one output for the adversary. Hence, the number of possible output vectors is at most the number of databases, i.e. $|S| \leq 2^{k\gamma}$. Algorithm $\mathcal{T}$ is also very simple. Given an output vector $\vec{O}$, it checks whether it is consistent for some database and in each session, the database entries in the output vector satisfy the adversary's input predicate.

In this work, we give a $O(n^\epsilon)$ round protocol (for any constant ϵ) in contrast with $O(n^3k^2)$ protocol by [Goy12] for this functionality. More precisely,

Theorem 4 *Assume the existence the collision resistant hash functions and constant-round semi-honest oblivious transfer. Then for any constant ϵ, there exists a $O(n^\epsilon)$ round real world protocol which realizes the ideal world experiment for private database search functionality.*

As also pointed out in [Goy12], several other problems of interest (such as private information retrieval, pattern matching, etc) are instances of the general problem of private database search discussed above. Arguments similar to those used for private database search can be used to show that other problems such as secure set intersection, computing k-th ranked element, etc also satisfy the bounded pseudoentropy condition.

Password Checking. Consider the following password checking functionality where a single honest party might interact with several adversarial parties in an unbounded number of concurrent sessions. The honest party holds a password p and the ideal functionality is such that if both parties input the same password, it outputs 1 to the adversarial parties, otherwise it outputs $\perp$. [Goy12] showed that this functionality satisfies the key technical property for $D = 2$. By lemma 1, it satisfies BPC for $B = \log^2 n + 1$.

Another way of interpreting this bound is the following: Let the number of sessions be m and let number of distinct inputs used by the adversary be $\tilde{m}$. Then the total number of possible output vectors are at most $\tilde{m} + 1$ because either none of the inputs match or exactly one of them matches. Since there are only a polynomial number of sessions, $B = \log^2 n + 1$ suffices. The algorithm $\mathcal{T}$ is very simple. For any output vector, it would accept if the output is 1 corresponding to at most one value of input, otherwise it rejects.

Bounded Concurrent MPC. Now consider the setting where there is an a-priori fixed bound k on the total number of concurrent sessions in the ideal world (see, e.g., [Lin03a, PR03, Pas04]). KTP is trivially satisfied for $D = k + 1$. Let γ be the length of output in any session. It is easy to see that this ideal world satisfies BPC for $B = k\gamma$. The algorithm $\mathcal{T}$ is trivial and accepts all the output vectors of length $k\gamma$.

D.1 Blind Signatures.

Next, we consider the application of blind signature introduced by [Cha82]. A blind signature scheme BS is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{User}, \mathsf{Verify})$. KeyGen is the key generation algorithm using which the signer generates the signing key and the verification key as $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KeyGen}(1^n)$. If the user wants to obtain a signature on some message Msg, the user and the signer engage in a signing protocol defined by algorithms $\mathsf{Sign}(\mathsf{sk})$ and $\mathsf{User}(\mathsf{Msg})$. The verification algorithm $\mathsf{Verify}(\mathsf{Msg}, \sigma, \mathsf{vk})$ outputs 1 is the signature σ is valid and 0 otherwise.

A blind signature scheme BS needs to satisfy *blindness* and *unforgeability*. BS satisfies blindness if any PPT adversarial signer cannot distinguish between the executions of signing protocol for any two distinct messages. BS satisfies unforgeability if any PPT adversarial user cannot output $k + 1$ valid message/signature pairs after interacting in k executions of the signing protocol.

Consider a signer with input sk and a user with input Msg to be signed. Then we can model blind signature as a two party computation between the signer and the user for the circuit for generating signatures. Note that the circuit will have the verification key vk hardcoded. At the end of the protocol, the user outputs a valid signature σ if obtained or $\perp$ otherwise. There is no

output of the signer, i.e. signer always outputs $\perp$. Also, consider the standard simulation based security definition.

The question whether concurrently secure protocol for blind signatures can be constructed as per the ideal/real model simulation paradigm was open till now. Lindell [Lin08] showed that any protocol that computes the blind signature in the plain model and remains secure for m concurrent executions, where security is proven via black-box simulation, must have at least m rounds of communication. This result motivated a number of results for concurrently secure blind signatures either using set up assumptions like common reference string [Fis06, KZ06, Oka06], random oracles [Bol03] or weaker notions of security (like game based security or synchronized attacks) [Oka06, HKKL07, GRS$^+$11, GG14].

The black box impossibility result by Lindell [Lin08] implies that non-black box techniques are essential to get a concurrently secure protocol for blind signatures. In this work, we give a positive result for concurrently secure blind signatures in the plain model using non-black-box simulation techniques. In particular, we show the following

Theorem 2 *(Restated) Assume the existence of collision resistant hash functions and constant-round semi-honest oblivious transfer. Then for any constant ϵ, there exists a $O(n^\epsilon)$ round real world protocol which realizes the ideal world for blind signature functionality.*

We prove this theorem by using unique signatures as the underlying signature scheme and showing that blind signatures satisfy the bounded pseudoentropy condition when the underlying signature scheme is unique. We first define unique signatures introduced by Goldwasser and Ostrovsky [GO92]. A signature scheme is said to be *unique* if for each public key pk and each message Msg, there exists at most one signature σ such that $\mathsf{Verify}(\mathsf{Msg}, \sigma, \mathsf{vk}) = 1$. Note that the black-box impossibility result of Lindell [Lin08] also holds for blind signature schemes with underlying unique signatures.

Now we show that the blind signature scheme which uses unique signatures satisfies bounded pseudoentropy condition. In particular, we show this for $B = 0$ and $\mathcal{T}$ algorithm which is same as the verification algorithm Verify. Note that if the adversary is playing the role of the user, its output is unique and is completely determined by its input Msg since vk is fixed by the function being computed. If the adversary is playing the role of the signer, its output is always $\perp$. Hence, set S will contain only one output vector, which is information theoretically fixed by the adversary inputs and the ideal world experiment, which fixes the verification keys for all the sessions. The algorithm $\mathcal{T}$ works as follows: It accepts the output vector if the signatures, for the sessions where the adversary is the user, verify under the corresponding public key and output is $\perp$ for the sessions where the adversary is the signer.

Finally note that blind signatures will not satisfy the key technical property. Consider the case when the adversary is acting as the user in all the sessions. By the unforgeability property of the scheme, any PPT predictor which receives k valid input/output (message/signature) pairs cannot predict the signature on the next message by non-negligible probability. Also, note that blind signatures will not satisfy the generalized key technical property discussed in the full version [Goy11] for the same reason.

D.2 Verifiable Random Functions.

Verifiable random functions (VRFs) were introduced by Micali, Rabin, and Vadhan [MRV99]. They combine the properties of pseudo-random functions with the verifiability property. Intuitively, they are like a pseudo-random function with a public key and proofs. Parts of the following description have been taken verbatim from [Lys02].

A function family $F_{(\cdot)}(\cdot) : \{0,1\}^n \to \{0,1\}^{n(k)}$ is a verifiable random function (VRF) if there exist probabilistic algorithm G, and deterministic algorithms Eval and Prove, and algorithm Verify such that $G(1^n)$ generates key pairs PK, SK; $\mathsf{Eval}(SK, x)$ computes the value $y = F_{PK}(x)$; $\mathsf{Prove}(SK, x)$ computes a proof π that the value $y = F_{PK}(x)$, and $\mathsf{Verify}(PK, x, y, \pi)$ verifies that $y = F_{PK}(x)$ using the proof π.

Additionally a VRF should satisfy the following properties: (1) Pseudo-randomness of $F_{PK}(x)$, i.e. based purely on PK, oracle calls to $F_{PK}(\cdot)$ and corresponding proof oracle, no PPT adversary can distinguish the value $F_{PK}(x)$ from random without explicitly querying the oracle on x. (2) Uniqueness, i.e. there do not exist values $(PK, x, y_1, y_2, \pi_1, \pi_2)$ such that $y_1 \neq y_2$ and $\mathsf{Verify}(PK, x, y_1, \pi_1) = \mathsf{Verify}(PK, x, y_2, \pi_2) = \mathsf{Accept}$.

VRFs are useful tools for protocol design. They can be viewed as a commitment to an exponential number of random looking bits, which can be used in protocols. For example, it is shown in [MR01] that using VRFs one can reduce the number of rounds for resettable zero-knowledge proofs to 3 in the bare model. Another example application, due to [MR02], is a non-interactive lottery system used in micro-payments.

Now consider the two party setting where one party has the secret key and other party wants to compute the corresponding function F_{PK} on input x of its choice. Output of the first party is $\perp$ and the second party party is y, π such that $\mathsf{Verifiy}(PK, x, y, \pi) = \mathsf{Accept}$. We view this as two party computation of a circuit corresponding to F_{PK}. We give a positive result for the ideal world setting where unbounded number of such sessions may be running concurrently even with interchangeable roles.

Theorem 3 *(Restated) Assume the existence of collision resistant hash functions and constant-round semi-honest oblivious transfer. Then for any constant ϵ, there exists a $O(n^\epsilon)$ round real world protocol which realizes the ideal world experiment for verifiable random functions.*

We prove this theorem by showing that VRFs satisfy the bounded pseudoentropy condition. In particular, we show this for $B = 0$ and $\mathcal{T}$ algorithm same as verification algorithm Verify. Note that if the adversary is playing the role of the second party, its output is unique and is completely determined by its input x, since both Eval and Prove are deterministic algorithms for any PK. If the adversary is playing the role of the first party, its output is always $\perp$. Hence, set S will contain only one output vector, which is information theoretically fixed by the adversary inputs and the ideal world experiment. The algorithm $\mathcal{T}$ works as follows: It accepts the output vector if the following holds: for the sessions where the adversary is the second party, the output should verify under Verify and for the sessions where the adversary is the first party, the output is $\perp$.

Finally, note that VRFs will not satisfy the key technical property. Consider the case when the adversary is acting as the second party in all the sessions. A PPT predictor which succeeds in predicting the output of F_{PK} on x which was not queried before can be used to break the pseudo-randomness property of $F_{PK}(\cdot)$. Also, note that VRFs will not satisfy the generalized key technical property discussed in the full version [Goy11] for the same reason.

E Security Analysis

E.1 Soundness and Argument of Knowledge

In this section, we prove the following theorem.

Theorem 5 *The interactive argument system described in Section 3 is computationally sound. In fact, the interactive argument system is also an argument of knowledge.*

To prove this theorem we borrow ideas from [Goy13] but we face a couple of technical difficulties. Intuitively, we need to argue that no PPT cheating prover P^* can commit to a code such that it can output the challenge string r. This analysis was simpler in the earlier works [DGS09, Goy13] because the output of the program was fixed by the program itself. This is certainly not true in our case because we allow oracle queries of the type $\mathsf{output}(\cdot)$ whose queries do not fix the responses information theoretically. The main technical difficulty is how to handle these oracle queries since the adversary can use these to pass non-trivial information to the program Π, which can help him to predict the verifier challenge r. Our key idea is that there are only 2^B valid response vectors for any vector of such queries. Hence, the adversary can send at most B bits of additional information to the program Π. This is the place where we crucially use the fact that only a bounded number of outputs are accepted by the algorithm $\mathcal{T}$.

Another difficulty is that the adversary P^* can choose the statement to prove adaptively and does not declare it before the start of the protocol. Recall that the prover declares the statement after all the slots and UAs have completed. Hence, our soundness experiment is also over the coins of P^* which are used to define the statement x. The way we handle this issue is as follows: Intuitively, we prove that no PPT cheating prover can come up with a convincing UA execution irrespective of the statement x which it chooses. Now we use the soundness and argument of knowledge property of WIAOK to conclude that if an honest verifier accepts, then P^* should have committed to an actual witness to $x \in L$.

Finally, note that though the prover P and the verifier V are both PPT algorithms, we might need super-polynomial time to check the validity of the statement $x \in L$. This means that we cannot always check in polynomial time whether the soundness condition is violated by a malicious prover. But still the event of a cheating prover breaking the soundness condition is well-defined. Now we prove the above theorem in detail. Note that we prove this theorem for the $\langle P, V \rangle$ protocol which was defined w.r.t. to certain Λ, $\mathcal{T}$ and bound B. Parts of this proof have been taken verbatim from [Goy13].

Proof. (*of theorem 5*) Recall that in the protocol, after receiving h from the verifier, the (possibly malicious) prover sends $z = \text{COM}(h(\Pi))$ where Π could be any arbitrary program. We first analyze the probability of such a program being able to output the verifier random bit string r given the input $y_1 \in \{0,1\}^{|r|-B-n}$ and access to the two kinds of oracle queries which are answered using $y_2 \in \{0,1\}^{\leq n^{\log\log n}}$ and $y_3 \in \{0,1\}^{\leq n^{\log\log n}}$ as described in the specification of language Λ. Now when Π is executed, there are a number of possibilities of the output depending upon what the input y_1 is and how the oracle queries are answered. For the decommitment queries of the form $\mathsf{decommit}(\mathsf{str})$, which expect (r) in return such that $\mathsf{str} = \text{COM}(r)$, by the unique opening property of the commitment scheme COM (See Appendix A), the answer to the query is information theoretically fixed given the query itself. For any number of queries of the second kind, there can be at most 2^B possible valid vector of answers. Hence the input y_1 along with responses to second kind of queries information theoretically determine the output of Π. Hence, there are a total of

$2^{|r|-n}$ possible outputs of Π. Denote by X the set of these possible outputs. Now the probability of a string $r \in_R \{0,1\}^{|r|}$ being an element of this set is bounded by 2^{-n} which is negligible in n. The above argument still does not imply that $(h, z, r) \notin \Lambda$ since $z(= \text{COM}(h(\Pi)))$ does not information theoretically fix the program Π.

Let P^* be a malicious prover which adaptively chooses statements $x \notin L$ and is still able to successfully complete the protocol such that an honest verifier V outputs accept with a noticeable probability. We can assume P^* is deterministic without loss of generality. Then, there must exist indices i and j such the following happens with a noticeable probability ϵ: (a) P^* is able to successfully complete the protocol such that an honest verifier V outputs accept, *and*, (b) upon running extractor of the WIAOK system, we recover the transcript of the i-th universal argument which was simulating the j-th slot. We denote this event by complete-(i,j). Thus, probability of complete-(i,j) is ϵ. Note that this probability is taken over the coins of the entire experiment, i.e. all the coins of P^* and V and, in particular, the coins used to define the NP-statement x.

Now we define prefix1 and prefix2 of the protocol as follows. Call prefix1 to be the point where the prover has given a commitment z (i.e., has given the slot begin message) in the j-th slot and is now expecting the verifier message r. Call prefix2 to be point where the prover has given the first UA prover message of the i-th UA execution and is now expecting the verifier challenge.

Now it has to be the case that for atleast a fraction $\frac{\epsilon}{2}$ of prefix1 executions, the probability (over rest of the verifier random coins) that the event complete-(i,j) happens is atleast $\frac{\epsilon}{2}$. We call such prefix1 executions as good1. Furthermore, conditioned on prefix1 being good1, at least for $\frac{\epsilon}{4}$ fraction of prefix2 executions, the probability that the event complete-(i,j) happens is atleast $\frac{\epsilon}{4}$. We call such prefix2 executions as good2.

Now the verifier executes in two phases as follows. In phase 1:

- The verifier honestly executes the full protocol with the prover and runs the extractor associated with the WIAOK system.
- If the event complete-(i,j) happens, the verifier rewinds the adversarial prover to prefix2 (i.e., the point where the prover expects the UA challenge in the i-th UA execution). The verifier aborts otherwise.

  Note that due to rewinding to prefix2, the statement x being proven by P^* may change. But the UA first message in i-th execution won't change.
- Now the verifier starts an external verifier of the 3-round universal argument system. By this point, the verifier has already extracted the opening of the commitment to the UA first prover message received in the i-th UA execution with the prover (as part of the witness extracted from the WIAOK system). The verifier forwards this UA first message to the external UA verifier.
- The verifier receives the UA challenge from the external UA verifier and simply passes it on to the prover.
- Now the verifier honestly completes the rest of the protocol with the prover and runs the extractor associated with the WIAOK system. If the event complete-(i,j) happens again, the verifier has extracted the last UA prover message (and aborts otherwise). It then forwards this message to the external UA verifier.

- The probability of this entire phase 1 getting complete without aborting is at least Pr[prefix1 is good1]· Pr[prefix2 is good2 | prefix1 is good1]·Pr[Event complete-(i, j) happens in two independent trials starting from prefix2]. This comes out to be $\frac{\epsilon}{2}\cdot\frac{\epsilon}{4}\cdot\frac{\epsilon}{4}\cdot\frac{\epsilon}{4}$ which is $\frac{\epsilon^4}{128}$.
- Now employing the weak knowledge extractor associated with the universal argument system [BG02], we can extract the witness from the UA with probability at least $p(\frac{\epsilon^4}{128})$ where p is a polynomial (recall that the probability of success of the extractor is polynomially related to the probability of success of the prover). This is because $\frac{\epsilon^4}{128}$ is the probability with which the external UA verifier gets a complete accepting UA execution. Thus, we have extracted a machine Π such that the prover's commitment z is a commitment to $h(\Pi)$.

Now in phase 2 of the experiment, the verifier rewinds the execution to prefix1 and completes the protocol execution honestly with fresh random coins (and in particular, giving a fresh challenge r to complete the slot). Rest of this phase is identical to phase 1. If the event complete-(i,j) happens, rewind the prover to prefix2, give the UA first message to the external UA verifier, get the UA challenge and pass it on to the verifier. Now the verifier honestly completes the rest of the protocol with the prover and if the event complete-(i,j) happens again, the verifier forwards the UA last message to the external UA verifier. The knowledge extractor associated with the UA system is again employed to recover the UA witness.

Note that conditioned on prefix1 being good1, phase 2 is completed without aborting with probability at least $\frac{\epsilon}{4}\cdot\frac{\epsilon}{4}\cdot\frac{\epsilon}{4}$ which is $\frac{\epsilon^3}{64}$. Hence, with probability at least $p(\frac{\epsilon^3}{64})$, we have extracted another machine Π' such the prover commitment z is a commitment to $h(\Pi')$

Now observe that the fresh challenge r (in phase 2) was chosen by the verifier after it received the program Π in phase 1. As argued in the previous paragraph, if X_Π is the set of all possible outputs of Π, the probability that $r \in X_\Pi$ is negligible. If phase 2 succeeds, the verifier has obtained another program Π'. As argued before, except with negligible probability, Π could not have predicted r and hence $\Pi \neq \Pi'$. However since $h(\Pi) = h(\Pi')$, we have obtained a collision in the hash function. The probability of this event COLL can be computed as follows:

$$\begin{aligned}\Pr[\text{COLL}] &\geq \Pr[\text{Phase 1 succeeds in extracting } \Pi]\cdot\Pr[\text{Phase 2 succeeds in extracting } \Pi'] \\ &\quad -\Pr[\Pi = \Pi'] \\ &\geq p(\frac{\epsilon^4}{128})\cdot p(\frac{\epsilon^3}{64}) - \text{negl}(n)\end{aligned}$$

which is still noticeable. This violates the collision resistance property of the function family $\mathcal{H}$.

This means that for every i, j, the probability of the event complete-(i,j) occurring is negligible. This must mean that the extractor of the WIAOK instead outputs a witness for $x \in L$ which was committed in the NMCom phase. This proves soundness as well as argument of knowledge property. ■

Following lemma will be useful in proving the simulation soundness of $\langle P, V\rangle$.

Lemma 2 *For any PPT prover P^* running an execution of $\langle P, V\rangle$ with an honest verifier such that the verifier accepts, the following holds: The commitment given in the* NMCom *phase is a valid commitment to a witness for $x \in L$, which is the statement being proved by P^*, with all but negligible probability.*

Proof. This lemma follows by soundness and argument of knowledge property of WIAOK. As shown above, the probability that the extractor of WIAOK outputs a transcripts of an accepting UA is negligible for any PPT P^*. Hence, the extractor always outputs a witness for $x \in L$ which was committed in the NMCom phase. This proves the lemma. ■

E.2 Description of the Simulator

Let there be k parties in the system where different pairs of parties are involved in one or more sessions of $\langle P, V \rangle$, such that the total number of sessions is polynomial in the security parameter n. A party might be playing the role of the prover in one session and the role of the verifier in other session, i.e. we are in the setting of interchangeable roles. Let $\mathcal{A}$ be an adversary who controls an arbitrary number of parties. For simplicity of exposition, we will assume that exactly one party is corrupted in each session. We note that if the real and ideal distributions are indistinguishable for this case, then by using standard techniques we can easily remove this assumption.

Below we describe a simulator for zero-knowledge which works for any given language Λ. In particular, it works any given algorithm $\mathcal{T}$ and bound B (see Figure 1). Since we are in the case when only zero-knowledge protocols are running concurrently, there are no calls to the trusted party in the ideal world and the adversary is not getting any non-trivial information in the ideal world. In other words, no additional information needs to be communicated to the adversary. Looking ahead, our simulator will not make any queries of the type output$(\cdot)$. Thus, our simulator trivially works for all possible languages Λ.

Furthermore, let n^c be the total number of messages across all concurrent sessions. Let $Mark()$ be the marking strategy, the combinatorial module used to mark a UA prover message as blank or simulate when the prover is honest as defined in [Goy13] (see Section 4 in [Goy13]). Note that this procedure also describes how to pick a slot for simulation for a UA which is marked simulate . [Goy13] showed that this marking strategy picks at least one UA in each session for simulation (coverage property) and the slots are picked for simulation such that the simulator runs in polynomial time. We refer the reader to [Goy13] for more details. For describing our simulator, we borrow ideas from [Goy13] and parts of the following description has been taken verbatim from [Goy13].

Description of $\mathcal{S}$. As in [Goy13], the simulator will be divided into two parts: one referred to as *the Oracle*, and, the other referred to as *the next message machine* (or NMM in short). Recall that the entire transcript has exactly n^c messages. We assume that the random tape of the NMM is divided into n^c strings of equal lengths (a pseudo-random generator may be applied on a string to expand it, if required). The i-th string will be used as randomness to generate the i-th message of the transcript (or verify the i-th message in case it is an incoming message). Denote these string by $s_1, \ldots, s_{n^c}$.

As the first step, *the Oracle generates these strings* and produces n^c commitments to these strings denoted by $C_1, \ldots, C_{n^c}$. The strings as well as the randomnesses used to generate these commitments are retained by the Oracle while the commitments themselves are given to the NMM (which then stores these commitments). Denote by $C = (C_1, \ldots, C_{n^c})$. The combinatorial module $Mark()$ which is used to mark a UA prover message of an honest party either blank or simulate is a part of the NMM and has its own random tape s. The string s is generated and stored by the NMM.

Throughout the simulation, the NMM will not have the randomness required to compute the next message. This critical missing information is stored by the Oracle. To compute the next outgoing message, the NMM may access the Oracle through a clearly defined interface: make a call to the oracle by producing a query of the form $\mathsf{decommit}(\mathsf{str})$ and expecting (r) such that $\mathsf{str} = \textsc{com}(r)$ in return. Hence, the NMM slowly learns the required random tapes and the associated decommitment information as we proceed in the simulation.

The intuition behind such a separation of the simulator into two parts will only be clear once we go through the hybrid experiments. However, following is a rough justification. To complete the simulation, the simulator will be required to commit to a machine which can predict the challenge r of the adversary. One option is to just commit to the entire state of the simulator (and the adversary) as the machine. However, if all the randomness for all the messages is committed, the changes to various messages required as part of the hybrid argument seem difficult to perform. To solve this issue, we commit to a machine which does not have these random tapes for the future messages. This machine can still regenerate the required transcript with the help of an external Oracle. The next message machine NMM represents the part of the simulator that will be committed as the machine while the Oracle represents the external uncommitted information that will be available to this machine while trying to regenerate the transcript.

The NMM handles each outgoing message as described below. In each of these steps, the randomness required by the NMM is taken from the string s_i (assuming that the message will be the i-th message exchanged between the adversary and the NMM). Indeed, the NMM does not have the required string s_i but only a commitment c_i to it. To get s_i, it just queries the Oracle with c_i.

Prover Messages:

1. **The slot begin message z:** The NMM defines a machine Π which simply consists of all the information the NMM has as part of its own state plus the current state of the adversary. However, the machine Π does not have the information that the Oracle stores (which is not yet learnt by the NMM). The NMM then computes $z = \textsc{com}(h(\Pi))$ and sends z to the adversary.

2. **A UA message marked blank :** The NMM runs the combinatorial module to see if the UA message is marked blank . If so, the NMM simply generates a random string of appropriate size and sends a commitment to that string to the adversary.

3. **A UA message marked $\mathsf{simulate}$:** If the combinatorial module marks a UA prover message as $\mathsf{simulate}$, the NMM proceeds as follows. If it is a UA first message, the NMM constructs a witness for the statement $(h, z, r) \in \Lambda$ and uses that to compute the UA first message (details of how the witness is constructed is given later on). The NMM now commits to the resulting UA first message. The resulting UA first message and the opening to the commitment are now stored by the NMM.

   Note that the NMM does not store any other intermediate information resulting from this step. In particular, it discards the witness and other information required to compute the UA first message (such as the PCP). This is done to ensure that the state of the NMM (and hence the machine Π it commits to when a slot begins) is bounded by a fixed polynomial. This is crucial to keep the simulation polynomial time.

   If the message is a last UA prover message such that the corresponding UA first message was also marked $\mathsf{simulate}$, then the NMM *again reconstructs* the witness for the appropriate

statement $(h, z, r) \in \Lambda$, reconstructs the UA first message, and then, computes the UA last message depending upon the UA challenge given by the adversary. Now the NMM computes a commitment to this UA last message and sends it to the adversary. The computed message and the opening to the commitment are again stored by the NMM. Note that, as earlier, the NMM discards any intermediate information computed in this step.

If the message is a UA last message where the UA first message was marked blank , the NMM simply generates a random string of appropriate size and sends a commitment to that string to the adversary.

4. **Non-Malleable Commitment:** The NMM commits to an all zero string (instead of an actual witness to x) using the non-malleable commitment scheme NMCom and the identity of the honest prover.

5. **Witness-indistinguishable argument of knowledge:** The NMM uses a witness to the statement: "there exists i such that the i-th UA execution was convincing". By the coverage property of the marking strategy, before we reach the WI stage of a session, there must exist an i such that both the UA first and last messages were marked simulate except with negligible probability (if this doesn't happen, the simulation is aborted). This means that for that i, the NMM must have computed the UA first and last message such that the UA execution becomes convincing. A witness to this statement is simply the openings to the commitments to the UA first and last messages, which is a part of the state of the NMM.

   Observe that, by the properties of the UA system, the computation required in this step is similar for each session and, in particular, does not "grow" as the simulation proceeds (since it depends only on the computational complexity of the UA verifier and not on that of the UA prover).

Constructing the witness. Now we show that for any universal argument which is selected for simulation, there exists a witness for the statement $(h, z, r) \in \Lambda$. This can be seen by construction of the NMM and the Oracle. We observe the following:

- Consider the point till which the NMM had generated the message z in the transcript. From this point onwards, the NMM, given only queries to the Oracle and access to $\mathcal{A}$, was able to continue generating the transcript and arrive at the challenge r. Thus, the machine Π can perform the same execution since it starts from the same internal state *assuming* it gets access to the same Oracle query responses as well.

- Now consider the point when the first UA message is due which is marked simulate . Let the index of the challenge message r, for the slot picked for simulation, in the transcript be j and set $y_1 = j$. Note that $|y_1|$ is $\mathcal{O}(\log n)$ and hence, $|y_1| \leq |r| - n - B$. Furthermore, let y_2 contain all the Oracle queries the NMM made while going from the message z to r (both inclusive) and the corresponding oracle responses. Now observe that any Oracle queries Π makes (before it outputs r) can be answered using y_2. This is because its execution would be identical to that of the NMM. Also note that our simulator for zero-knowledge will not make any oracles calls of kind output$(\cdot)$.

- Now the witness simply consists of Π, y_1, y_2, and, the opening of the commitment z. The NMM already has all of these as part of its internal state by the time it reaches r (and in

particular, by the time the first UA message is due). Thus, the NMM has computed the witness required to complete the universal argument.

Verifier Messages: NMM generates all the honest verifier messages using honest strategy with a small modification. It generates the honest verifier challenges by applying a suitably expanding pseudorandom generator to the randomness given for the message.

E.3 Indistinguishability of Views and Simulation Soundness

Informally speaking, in this section we will prove the following two properties for our simulation.

Indistinguishability of Views. For any PPT adversary $\mathcal{A}$ (described above), the real and the simulated executions are computationally indistinguishable.

Simulation Soundness. Each proof given by the adversary $\mathcal{A}$ is sound even if it is receiving one or more simulated proofs.

In this section, we borrow ideas from [GGJS12, Goy13]. In fact, parts of the text has been taken verbatim from [GGJS11] or [Goy13].

We consider two experiments $\mathcal{H}_{\mathsf{Real}}$ and $\mathcal{H}_{\mathsf{Ideal}}$, where $\mathcal{H}_{\mathsf{Real}}$ corresponds to the real concurrent executions of $\langle P, V\rangle$ while $\mathcal{H}_{\mathsf{Ideal}}$ corresponds to the simulated executions, as described above (Section E.2).

Experiment $\mathcal{H}_{\mathsf{Real}}$: The simulator $\mathcal{S}$ is given the witnesses for all the sessions with honest prover and adversarial verifier. It interacts with the adversary following the honest party strategies in all the sessions. The output of the hybrid corresponds to the view of the adversary $\mathcal{A}$ and the outputs of the honest verifiers.

Experiment $\mathcal{H}_{\mathsf{Ideal}}$: $\mathcal{S}$ simulates all the sessions without the witnesses of the honest provers (in the same manner as explained in the description of $\mathcal{S}$ in Section E.2) and outputs the view of $\mathcal{A}$. Again the output of the hybrid corresponds to the view of the adversary $\mathcal{A}$ and the outputs of the honest verifiers.

Let ν_i be a random variable that represents the output of $\mathcal{H}_i$. We now claim that the output distributions of $\mathcal{H}_{\mathsf{Real}}$ and $\mathcal{H}_{\mathsf{Ideal}}$ are indistinguishable, as stated below:

Lemma 3 $\nu_{\mathsf{Real}} \stackrel{c}{\equiv} \nu_{\mathsf{Ideal}}$

We will prove this lemma using a carefully designed series of intermediate hybrid experiments. More details are given below.

Soundness Invariant. While proving the indistinguishability of the outputs of our hybrid experiments, we will also argue that in each session where $\mathcal{A}$ acts as the prover, the commitment in the NMCOM phase is a valid commitment to the witness w for the statement x being proven. We refer to this as the *soundness condition*.

Let the number of sessions where the adversary acts as the prover be m. We denote by $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ the ℓth instance of $\langle P, V\rangle$, where $\mathcal{A}$ is the prover. Let $\pi^{\ell}_{\mathcal{A}}$ denote the statement being

proven in $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$. Let ρ^{ℓ} denote value committed in NMCOM phase in $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$. Note that the soundness condition "holds" if we prove that in each session $\ell \in [m]$, $\mathcal{A}$ commits to a valid witness to the statement $\pi^{\ell}_{\mathcal{A}}$, i.e. ρ^{ℓ} is a valid witness to $\pi^{\ell}_{\mathcal{A}}$.

Now, before we proceed to the description of our hybrids, we claim that the soundness condition holds in the real execution. We will later argue that the soundness condition still holds as we move from one hybrid to another. We call this *soundness invariant.*

Lemma 4 *Let $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ and $\pi^{\ell}_{\mathcal{A}}$ be as described above corresponding to the real execution. Then, for each session $\ell \in [m]$, if the honest verifier accepts $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$, then ρ^{ℓ} is a valid commitment to a witness of $\pi^{\ell}_{\mathcal{A}}$ except with negligible probability.*

The above lemma immediately follows the lemma 2 for the stand alone setting (Section E.1).

Lemma 5 *Let $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ and $\pi^{\ell}_{\mathcal{A}}$ be as described above corresponding to the simulated execution in $\mathcal{H}_{\mathsf{Ideal}}$. Then, for each session $\ell \in [m]$, if the honest verifier accepts $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$, then ρ^{ℓ} is a valid commitment to a witness of $\pi^{\ell}_{\mathcal{A}}$ except with negligible probability.*

Note that this lemma implies simulation soundness. We will prove this lemma using a carefully designed series of hybrids.

E.3.1 Getting Started

We will prove Lemma 3 and Lemma 5 by contradiction. Suppose that the hybrids $\mathcal{H}_{\mathsf{Real}}$ and $\mathcal{H}_{\mathsf{Ideal}}$ are distinguishable in polynomial time, i.e., there exists a PPT distinguisher D that can distinguish between the two hybrids with non-negligible probability. We will now consider a series of hybrid experiments $\mathcal{H}_{\mathsf{first}}$ followed by $\mathcal{H}_{i:j}$, where $i \in [n^c]$, and $j \in [4]$ and $\mathcal{H}_{\mathsf{last}}$. For each intermediate hybrid $\mathcal{H}_k$, we define a random variable ν_k that represents the view of $\mathcal{A}$ in $\mathcal{H}_k$.

Below, we show that no PPT distinguisher can distinguish between any two consecutive hybrids, which will establish (via the hybrid arguments) that no polynomial time distinguisher can distinguish between ν_{Real} and ν_{Ideal} with a non-negligible probability, which is a contradiction. Additionally, we will prove that soundness invariant holds in the current hybrid assuming that it holds in the previous hybrid. This will prove lemma 5.

Proving lemma 5 is the main non-trivial part of this section. We need to ensure that as we change the proofs given to the adversary from being correct (using the witness) to simulated, the adversary continues to commit to the actual witness to the statement being proven. This requires certain "non-malleability" from the $\langle P, V\rangle$ protocol. For this, we will rely on the k-robustness and non-malleability of the commitment scheme NMCom. More intuition about the proof is given later.

In the rest of the section, we will denote by $\langle P, V\rangle_{H\to\mathcal{A}}$ the instances where an honest prover is giving a proof to $\mathcal{A}$ and by $\langle P, V\rangle_{\mathcal{A}\to H}$ the instances where adversary is proving statements of his choice to honest verifier.

E.4 Description of the Hybrids

We describe the hybrids $\mathcal{H}_{\mathsf{first}}$, $\mathcal{H}_{i:j}$ for $i \in [n^c]$ and $j \in [4]$ and $\mathcal{H}_{\mathsf{last}}$ below:

Experiment $\mathcal{H}_{\mathsf{first}}$: In this experiment, the simulator is divided into the Oracle and the next message machine (NMM). The Oracle generates the strings $s_1, \ldots, s_{n^c}$, computes the commitments $C = (C_1, \ldots, C_{n^c})$ and gives C to the NMM. The oracle also has the witnesses for all the sessions with honest provers and adversarial verifiers and computes $c_{w_i} = \text{COM}(w_H^i)$ as commitment of witness in i-th session. It gives $C_w = (c_{w_1}, c_{w_2}, \ldots, c_{w_m})$ to NMM, where m are the number of sessions with honest provers. In addition, the NMM generates randomness s required to run the combinatorial module. To compute the next message, the NMM queries the Oracle to get the appropriate random string and honest party witness for that session (if required) and computes the required message behaving honestly. Furthermore, the NMM runs the combinatorial module (using the random tape s) and internally marks each UA message either simulate or blank . The only change between this hybrid and $\mathcal{H}_{\mathsf{Real}}$ is that the NMM uses a PRG to generate the verifier challenge strings in all sessions of $\langle P, V \rangle_{\mathcal{A} \to H}$. We claim that

$$\nu_{\mathsf{Real}} \stackrel{c}{\equiv} \nu_{\mathsf{first}} \tag{1}$$

$$\forall \ell \quad \rho^{\ell}_{\mathsf{Real}} \stackrel{c}{\equiv} \rho^{\ell}_{\mathsf{first}} \tag{2}$$

Proving Equation 1. To prove this, we will consider another sequence of hybrids, in which we change the honest verifier challenges one by one from being random to pseudorandom. Let $\mathcal{H}_{\mathsf{Real}:1}$ be $\mathcal{H}_{\mathsf{Real}}$. For $1 \leq j \leq p$, where p are the total number of honest verifier challenge messages, consider two consecutive hybrids $\mathcal{H}_{\mathsf{Real}:j}$ and $\mathcal{H}_{\mathsf{Real}:(j+1)}$ such that first $(j-1)$ verifier challenges are pseudorandom in both $\mathcal{H}_{\mathsf{Real}:j}$ and $\mathcal{H}_{\mathsf{Real}:(j+1)}$, and the jth verifier challenge in the transcript is random in $\mathcal{H}_{\mathsf{Real}:j}$ and pseudorandom in $\mathcal{H}_{\mathsf{Real}:(j+1)}$. Rest all verifier challenges will be random in both $\mathcal{H}_{\mathsf{Real}:j}$ and $\mathcal{H}_{\mathsf{Real}:(j+1)}$. Now, this j-th verifier challenge will not be generated by the NMM. Instead, it will be given to it externally. Moreover, the NMM will not receive any randomness for this message from the oracle. The indistinguishability of $\nu_{\mathsf{Real}:j}$ and $\nu_{\mathsf{Real}:(j+1)}$ will hold by the security of the PRG. Equation 1 holds by a standard hybrid argument.

Proving Equation 2. Let $\mathcal{H}_{\mathsf{Real}:j}$ and $\mathcal{H}_{\mathsf{Real}:(j+1)}$ be as above. Let us assume there exists a ℓ such that $\rho^{\ell}_{\mathsf{Real}:j}$ and $\rho^{\ell}_{\mathsf{Real}:(j+1)}$ are distinguishable by a PPT distinguisher D. We will create a standalone machine M^* that is identical to $\mathcal{H}_{\mathsf{Real}:j}$, except that it takes this verifier challenge externally from B and forwards it to $\mathcal{A}$. Additionally, M^* exposes the NMCom inside $\langle P, V \rangle^{\ell}_{\mathcal{A} \to H}$ to an external receiver R. If B sends a random string, then (B, M^*, R) system is identical to $\mathcal{H}_{\mathsf{Real}:j}$. On the other hand, if B sends a pseudorandom string, then (B, M^*, R) system is identical to $\mathcal{H}_{\mathsf{Real}:(j+1)}$. Since only one message is changing on the left, by security of PRG and the k-robustness of NMCom, the value committed by M^* to R is indistinguishable in two cases. Equation 2 holds by a standard hybrid argument.

Intuition behind rest of the proof. Next, we shall consider n^c sets of intermediate hybrid experiments $\mathcal{H}_{i:1}, \mathcal{H}_{i:2}, \mathcal{H}_{i:3}, \mathcal{H}_{i:4}$ for $i \in [n^c]$. For each of the hybrids, we will prove that the view of $\mathcal{A}$ in the current hybrid is indistinguishable from the previous hybrid. We will also prove that the soundness condition holds in the current hybrid assuming that it holds in the previous hybrid.

Intuition behind indistinguishability proof. The basic problem we deal with in these hybrid experiments occurs because of non-black-box simulation. We are using some cryptographic primitives in our protocol such as commitment schemes COM and NMCom (providing hiding property)

and a WIAOK system. We would go through the hybrids by changing what we commit to (and then rely on the hiding property of the commitment scheme) and by changing the witness we use in the WIAOK system (rely on the witness indistinguishability).

Now, say that the NMM starts using the opening of the commitment C_i (to the string s_i) while either preparing the machine it commits to (as part of the slot begin message), or, while completing a simulated universal argument. Then, if a primitive is executed using randomness s_i, we note that its security properties (which we rely on), may no longer hold. For example, if the commitment is prepared using randomness s_i, we can no longer rely on the hiding property of this commitment.

To solve this problem, we shall move forward in the transcript and make changes in a sequential manner. When we are in the i-th set of hybrids, we will maintain the following invariants: (a) the NMM will only use the opening of the commitments C_j for $j < i$, and, (b) from this hybrid onwards, we will only change messages indexed i and higher. We must have already made all the required changes in messages before the i-th message.

Intuition behind soundness invariant proof. To argue the soundness of the current hybrid assuming the soundness condition for the previous hybrid, we will rely on the *k-robustness* of NMCom (Appendix A.6). In each hybrid, we change at most k messages in a session on the left (i.e. a session with honest prover) amd argue that the value inside the non-malleable commitment given by the adversary cannot change. In a hybrid where we change a non-malleable commitment on the left, we will rely on the non-malleability of NMCom with respect to itself.

Define $\mathcal{H}_{0:4}$ to be the same as $\mathcal{H}_{\mathsf{first}}$.

Experiment $\mathcal{H}_{i:1}$: This experiment is identical to the previous one except in how the message with index i in the transcript is handled by the simulator. In this hybrid, if the i-th message is a slot begin message z, the NMM works as follows. Instead of committing to $h(0)$, the NMM now constructs a machine Π (which consists of the internal state of NMM and that of the adversary), computes the commitment $z = \text{COM}(h(\Pi); s_i)$ using the appropriate randomness s_i and sends it to the adversary.

We now claim that,

$$\nu_{i-1:4} \stackrel{c}{\equiv} \nu_{i:1} \tag{3}$$

$$\forall \ell \quad \rho^{\ell}_{i-1:4} \stackrel{c}{\equiv} \rho^{\ell}_{i:1} \tag{4}$$

Proving Equations 3 *and* 4. We claim that these hold by the (computational) hiding property of the commitment scheme COM and k-robustness of NMCom. The only reason it is non-trivial is that the string C contains a commitment C_i to the randomness s_i required to produce the commitment z. However, we argue that the opening to the commitment C_i is not used by the NMM in any non-trivial way in the interaction with the adversary and hence, the commitment generated by using s_i as randomness is semantically secure. We show this via a sequence of intermediate hybrids.

Experiment $\mathcal{H}_{i:1:1}$: This is identical to $\mathcal{H}_{i:1}$ except that the Oracle, upon receiving a query str, only returns value s.t. $\mathsf{str} = \text{COM}(\mathsf{value}; r)$ if $\mathsf{str} = C_j$ for $j \geq i$. Even given access to such a modified

Oracle, the NMM could produce a transcript which is identically distributed. We claim that

$$\nu_{i:1} \stackrel{s}{\equiv} \nu_{i:1:1} \tag{5}$$

$$\forall \ell \quad \rho^{\ell}_{i:1} \stackrel{c}{\equiv} \rho^{\ell}_{i:1:1} \tag{6}$$

Proving Equation 5. The openings given by the Oracle to the NMM are used by NMM only to either construct a witness for a simulated UA, or, to commit to a machine Π (because openings are part of the state of NMM) when a slot begins. However, there is no simulated UA or a commitment to a machine after the message i of the transcript in this experiment. Thus, for a message $j \geq i$, the NMM is not using the opening of the commitment C_j in any way in the experiment. Note that here too the NMM generates the commitment $z = \text{COM}(h(\Pi); s_i)$.

Proving Equation 6. Let us assume that equation 6 is false. That is, $\exists \ell \in [m]$ such that $\rho^{\ell}_{i:1}$ and $\rho^{\ell}_{i:1:1}$ are distinguishable by a probabilistic polynomial time (PPT) distinguisher. In this case, we can create an unbounded adversary that extracts the value contained in the non-malleable commitment and is then able to distinguish between the view of $\mathcal{A}$ in $\mathcal{H}_{i:1}$ and $\mathcal{H}_{i:1:1}$, which is a contradiction by equation 5.

Experiment $\mathcal{H}_{i:1:2}$: This is same as $\mathcal{H}_{i:1:1}$ except that we change the commitment C_i to be $\text{COM}(s'_i)$ such that s'_i is chosen independently. The oracle is identical to the previous one except on query $\mathsf{str} = C_i$, it responds with s_i defined before. We claim that

$$\nu_{i:1:1} \stackrel{c}{\equiv} \nu_{i:1:2} \tag{7}$$

$$\forall \ell \quad \rho^{\ell}_{i:1:1} \stackrel{c}{\equiv} \rho^{\ell}_{i:1:2} \tag{8}$$

Proving Equations 7 *and* 8. The first equation holds by the hiding property of COM and the second equation holds by k-robustness of NMCom.

We consider similar experiments $\mathcal{H}_{i-1:4:1}$ and $\mathcal{H}_{i-1:4:2}$ corresponding to $\mathcal{H}_{i-1:4}$ where the NMM generates the commitment $z = \text{COM}(0; s_i)$. We describe them in detail below.

Experiment $\mathcal{H}_{i-1:4:1}$: This is identical to $\mathcal{H}_{i-1:4}$ except that the Oracle, upon receiving a query str, only returns value s.t. $\mathsf{str} = \text{COM}(\mathsf{value}; r)$ if $\mathsf{str} = C_j$ for $j \geq i$. Even given access to such a modified Oracle, the NMM could produce a transcript which is identically distributed to that in $\mathcal{H}_{i-1:4}$. We claim that

$$\nu_{i-1:4} \stackrel{s}{\equiv} \nu_{i-1:4:1} \tag{9}$$

$$\forall \ell \quad \rho^{\ell}_{i-1:4} \stackrel{c}{\equiv} \rho^{\ell}_{i-1:4:1} \tag{10}$$

Proving Equations 9 *and* 10. The proof is exactly same as the proof for equations 5 and 6.

Experiment $\mathcal{H}_{i-1:4:2}$: This is same as $\mathcal{H}_{i-1:4:1}$ except that we change the commitment C_i to be $\text{COM}(s'_i)$ such that s'_i is chosen independently. The oracle is identical to the previous one except on query $\mathsf{str} = C_i$, it responds with s_i defined before. We claim that

$$\nu_{i-1:4:1} \stackrel{c}{\equiv} \nu_{i-1:4:2} \tag{11}$$

$$\forall \ell \quad \rho^{\ell}_{i-1:4:1} \stackrel{c}{\equiv} \rho^{\ell}_{i-1:4:2} \tag{12}$$

Proving Equations 11 *and* 12. The proof is exactly same as the proof for equations 7 and 8.

Note that to prove equations 3 and 4 it is sufficient to prove the following:

$$\nu_{i-1:4:2} \stackrel{c}{\equiv} \nu_{i:1:2} \tag{13}$$

$$\forall \ell \quad \rho^{\ell}_{i-1:4:2} \stackrel{c}{\equiv} \rho^{\ell}_{i:1:2} \tag{14}$$

Proving Equation 13. Note that in experiments $\mathcal{H}_{i-1:4:2}$ and $\mathcal{H}_{i:1:2}$, no information is revealed about the string s_i, which is used as randomness to generate the commitment z. Now NMM instead of generating the message z itself by querying the oracle for C_i, takes the message z externally under randomness s_i and does not make any query for C_i. If it receives a commitment to $h(0)$ then the experiment is identical to $\mathcal{H}_{i-1:4:2}$. If it receives a commitment to $h(\Pi)$, then the experiment is identical to $\mathcal{H}_{i:1:2}$. Since s_i is semantically secure, any commitment using s_i as randomness is also semantically secure. Thus, the experiments $\mathcal{H}_{i-1:4:2}$ and $\mathcal{H}_{i:1:2}$ are computationally close. Hence, the view of the adversary in experiments $\mathcal{H}_{i-1:4}$ and $\mathcal{H}_{i:1}$ is computationally close.

Proving Equation 14. It holds by the semantic security of commitment z and the k-robustness of NMCom. More precisely, let us first assume that the claim is false, i.e., $\exists \ell \in [m]$ such that $\rho^{\ell}_{i-1:4:2}$ and $\rho^{\ell}_{i:1:2}$ are distinguishable by a PPT distinguisher D. We will create a standalone machine M^* that is identical to $\mathcal{H}_{i-1:4:2}$, except that instead of simply committing to $h(0)$ using COM, M^* takes this commitment from an external sender B and "forwards" it internally to $\mathcal{A}$. Additionally, M^* "exposes" the NMCom in $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ to an external receiver R. This is feasible because we chose η to be more than communication complexity of receiver in NMCom plus the length of commitment z. Hence, both the receiver messages in NMCom and commitment z can be given externally to NMM.

Let us describe the interaction between M^* and B in more detail. M^* first sends the program Π to B. Now, when B sends the message z, M^* forwards this message to $\mathcal{A}$. Note that if B commits to $h(0)$ in the COM execution, then the (B, M^*, R) system is identical to $\mathcal{H}_{i-1:4:2}$. On the other hand, if B commits to the program $h(\Pi)$, then the (B, M^*, R) system is equivalent to $\mathcal{H}_{i:1:2}$. From the k-robustness of NMCom and hiding property of COM, we establish that the value committed by M^* to R must be computationally indistinguishable in both cases.

Experiment $\mathcal{H}_{i:2}$: This experiment is identical to the previous one except how the message with index i in the transcript is handled by the simulator. In this hybrid, if the i-th message is a UA message which is marked simulate , the way the NMM handles this message is identical to the final NMM (described earlier in Section E.2). In particular, the NMM starts using the witness guaranteed by the construction of Π for computing this UA message (as opposed to just committing to random strings). The intermediate information resulting from this step is discarded (as described earlier). We now claim that,

$$\nu_{i:1} \stackrel{c}{\equiv} \nu_{i:2} \tag{15}$$

$$\forall \ell \quad \rho^{\ell}_{i:1} \stackrel{c}{\equiv} \rho^{\ell}_{i:2} \tag{16}$$

Proving Equation 15. This relies on the computational hiding property of the commitment scheme COM using which the UA message is committed and is very similar to the previous indistinguishability proof. The difference between this hybrid and the previous one is only in the computation of the i-th message (in case it is a UA prover message marked as simulate). However, similar to

the previous indistinguishability argument, we can argue that the randomness used to compute the commitment to this UA prover message is not used by the NMM in anyway in the interaction with the adversary.

Proving Equation 16. After we prove that the randomness used to compute this commitment is not used by NMM anywhere, this equation follows by the hiding property of COM and k-robustness of NMCom in a way very similar to the proof of equation 4.

Experiment $\mathcal{H}_{i:3}$: This experiment is identical to the previous one except that now in all the sessions in which the preamble phase of $\langle P, V\rangle_{H\to\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript, the NMM starts using the alternative witness (the opening to the simulated UA messages) to complete the WIAOK. For any such session, up to (and including) the i-th message of the transcript, there exists a UA both of whose prover messages are marked simulate . The NMM now starts using the openings of these UA messages as the witness to complete the WIAOK execution. Note that there can be at most one session, in which the preamble phase of $\langle P, V\rangle_{H\to\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript and not completed upto (and including) the $(i-1)$th message of the transcript. That is there is only session where we are changing the witness in WIAOK compared to the previous hybrid. We now claim that,

$$\nu_{i:2} \stackrel{c}{\equiv} \nu_{i:3} \tag{17}$$

$$\forall \ell \quad \rho^{\ell}_{i:2} \stackrel{c}{\equiv} \rho^{\ell}_{i:3} \tag{18}$$

Proving Equation 17. This is very similar to the previous proof of indistinguishability. The only changes made in this hybrid occur after the message with index i in the transcript. However, as argued earlier, we can prove that the random strings used to compute the prover messages in WIAOK are still semantically secure. Then we can use an external prover to generate the messages of this WIAOK and give these as input to NMM. This can be done since η is chosen to be more than the length of prover messages in WIAOK. Hence, the string y_1 can include these messages from external prover of WIAOK. The claim follows by the witness indistinguishability of the WIAOK protocol.

Proving Equation 18. This is very similar to the proof of soundness condition in previous hybrids. Once we have proved that the randomness used to generate these prover messages of WIAOK are semantically secure, this equation follows by witness indistinguishability of the WIAOK protocol and k-robustness of NMCom. Since η is chosen to be greater than the communication complexity of prover messages of WIAOK and receiver messages of NMCom, we can take the prover messages of WIAOK externally and expose any of the commitments under NMCom given by the adversary to an external receiver R as above. Also, since k is chosen to be greater than 3, which is the round complexity of WIAOK, the indistinguishability of values committed by $\mathcal{A}$ follows by WI property of WIAOK and k-robustness of NMCom.

Experiment $\mathcal{H}_{i:4}$: This experiment in identical to the previous one except that now in all the sessions in which the preamble phase of $\langle P, V\rangle_{H\to\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript, the NMM instead of committing to actual witness in NMCOM phase, starts committing to an all zero string. Note that, as argued above, there is only one session where we are changing the value committed by NMCom compared to the previous hybrid. We now claim

that,

$$\nu_{i:3} \stackrel{c}{\equiv} \nu_{i:4} \tag{19}$$

$$\forall \ell \quad \rho^{\ell}_{i:3} \stackrel{c}{\equiv} \rho^{\ell}_{i:4} \tag{20}$$

Proving Equation 19. This is very similar to the previous proof of indistinguishability. The only changes made in this hybrid occur after the message with index i in the interaction transcript. However, as argued earlier, we can prove that the random strings used to compute this commitment are still semantically secure. Next, we can provide this commitment externally to the NMM since we set η more than the sender communication complexity of NMCom. The claim follows by the hiding property of NMCom.

Proving Equation 20. Once we have proved that the randomness used to generate these sender messages of NMCom are semantically secure, this equation follows by the non-malleability of NMCom w.r.t. itself. More precisely, let us assume that $\exists \ell \in [m]$ such that $\rho^{\ell}_{i:3}$ and $\rho^{\ell}_{i:4}$ are distinguishable by a PPT distinguisher D. We will create a standalone machine M^* that is identical to $\mathcal{H}_{i:3}$, except that instead of simply committing to witness using NMCom in $\langle P, V\rangle_{H\to\mathcal{A}}$, M^* takes this commitment from an external sender B and "forwards" it internally to $\mathcal{A}$. Additionally, M^* "exposes" the NMCom in $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ to an external receiver R. This can be done since η is chosen to be greater than the communication complexity of sender messages of NMCom plus the communication complexity of receiver messages of NMCom. Let us describe the interaction between M^* and B in more detail. M^* first sends the witness w to B. Now, when B starts the execution of NMCom in $\langle P, V\rangle_{H\to\mathcal{A}}$, M^* forwards the messages from B to $\mathcal{A}$; the responses from $\mathcal{A}$ are forwarded externally to B. Note that if B commits to the witness, then the (B, M^*, R) system is identical to $\mathcal{H}_{i:3}$. On the other hand, if B commits to a string of all zeros, then the (B, M^*, R) system is equivalent to $\mathcal{H}_{i:4}$. From the non-malleability property of NMCom, we establish that the value committed by M^* to R must be computationally indistinguishable in both cases.

This way we create the hybrids up to $\mathcal{H}_{n^c:4}$. Note that in this last hybrid, NMM does not use the honest prover witnesses to generate any message. Hence, it does not query the oracle with the decommitment for any c_{w_i}. So in the final hybrid, we remove the honest prover witnesses from the oracle. Intuitively, this is possible since oracle is the external uncommitted part of the simulator.

Experiment $\mathcal{H}_{\mathsf{last}}$: This hybrid is same as $\mathcal{H}_{n^c:4}$ except that now the oracle generates c_{w_i} differently. For each i, it picks a sufficiently long string r_i and computes $c_{w_i} = \text{COM}(r_i)$ and gives these commitments to NMM. We claim that

$$\nu_{n^c:4} \stackrel{c}{\equiv} \nu_{\mathsf{last}} \tag{21}$$

$$\forall \ell \quad \rho^{\ell}_{n^c:4} \stackrel{c}{\equiv} \rho^{\ell}_{\mathsf{last}} \tag{22}$$

Proving Equation 21. Since NMM in $\mathcal{H}_{n^c:4}$ never queries the oracle for decommitments of any c_{w_i}, the transcript in this hybrid is computationally indistinguishable from the previous hybrid by security of COM.

Proving Equation 20. To prove this we can consider a sequence of intermediate hybrids where we change the commitments c_{w_i} one by one. The computational indistinguishability of ρ^{ℓ} between

any two such consecutive hybrids will follow by the hiding property of COM and k-robustness of NMCom.

Note that $\mathcal{H}_{\mathsf{last}}$ is same as $\mathcal{H}_{\mathsf{Ideal}}$. Hence, we have proved that $\nu_{\mathsf{Real}} \stackrel{c}{\equiv} \nu_{\mathsf{Ideal}}$ and that the soundness of all the proofs given by the adversary holds in $\mathcal{H}_{\mathsf{Ideal}}$.

F Description of the Simulator

Let there be k parties in the system where different pairs of parties are involved in one or more sessions of Σ, such that the total number of sessions m is polynomial in the security parameter n. A party might be playing the role of P_1 in one session and P_2 in other session, i.e. we are in the setting of interchangeable roles. Let $\mathcal{A}$ be an adversary who controls an arbitrary number of parties. For simplicity of exposition, we will assume that exactly one party is corrupted in each session. We note that if the real and ideal distributions are indistinguishable for this case, then by using standard techniques we can easily remove this assumption. Now we first fix some notation.

NOTATION. In the sequel, for any session $i \in [m]$, we will use the notation H to denote the honest party and $\mathcal{A}$ to denote the corrupted party. Let $\langle P, V \rangle_{H \to \mathcal{A}}$ denote an instance of $\langle P, V \rangle$ where H plays the role of the prover and $\mathcal{A}$ plays the role of the verifier. Similarly, let $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle_{H \to \mathcal{A}}$ denote an instance of $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle$ where H and $\mathcal{A}$ plays the roles of prover and verifier respectively. Now, recall that the prover in $\langle P, V \rangle$ sends a non-malleable commitment to the other party. We will denote it by $\mathsf{com}_{H \to \mathcal{A}}$ when H plays the role of the committer. Further, we define $\langle P, V \rangle_{\mathcal{A} \to H}$, $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle_{\mathcal{A} \to H}$, $\mathsf{com}_{\mathcal{A} \to H}$ in the same manner as above, with the roles of H and $\mathcal{A}$ are interchanged. Also, let $x_{\mathcal{A}}$ and $r_{\mathcal{A}}$ denote the input and random coins, respectively, of $\mathcal{A}$ (to be used in the final secure computation phase).

Furthermore, let n^c be the total number of messages across all concurrent sessions. Let $Mark()$ be the marking strategy, the combinatorial module used to mark a UA prover message of an honest party as blank or $\mathsf{simulate}$ as defined in [Goy13]. Note that this procedure also describes how to pick a slot for simulation for a UA which is marked $\mathsf{simulate}$. [Goy13] showed that this marking strategy picks at least one UA in each session for simulation (coverage property) and the slots are picked for simulation such that the simulator runs in polynomial time. We refer the reader to [Goy13] for more details.

New Ideas. The description of the simulator builds upon the simulator of the simulation sound non-black-box concurrent zero-knowledge in Section E.2, but we face some technical difficulties. Unlike the adversary of zero-knowledge, the adversary in MPC learns outputs from the trusted functionality. In other words, there is additional information which needs to be communicated to the adversary during simulation. To handle this issue we crucially use the fact that the ideal world experiment satisfies the bounded pseudoentropy condition and hence there are at most 2^B output vectors which would be accepted by the algorithm $\mathcal{T}$ in language Λ. Another crucial point to note is that by the time the simulator needs to recreate the transcript of a slot (while computing the messages of a simulated UA), the outputs learnt by the adversary in that part of execution are already known to the simulator. Hence, it can use the oracle queries of the form $\mathsf{output}(\cdot)$ to give this crucial information, i.e. the outputs, to the machine which it committed to earlier.

Parts of the following description have been taken verbatim from [Goy13, GGJS11].

Description of $\mathcal{S}$. As in [Goy13], the simulator will be divided into two parts: one referred to as *the Oracle*, and, the other referred to as *the next message machine* (or NMM in short). We assume that the random tape of the NMM is divided into n^c strings of equal lengths (a pseudo-random generator may be applied on a string to expand it if required). The i-th string will be used as randomness to generate the i-th message of the transcript (or verify the i-th message in case it is an incoming message). Denote these string by $s_1, \ldots, s_{n^c}$.

As the first step, *the Oracle generates these strings* and produces n^c commitments to these strings denoted as $C_1, \ldots, C_{n^c}$. The strings as well as the randomnesses used to generate these commitments are retained by the Oracle while the commitments themselves are given to the NMM (which then stores these commitments). Denote by $C = (C_1, \ldots, C_{n^c})$. The combinatorial module $Mark()$, which is used to mark a UA prover message given to the adversary either blank or simulate , is part of the NMM and has its own random tape s. The string s is generated and stored by the NMM.

Throughout the simulation, the NMM will not have the randomness required to compute the next message. This critical missing information is stored by the Oracle. To compute the next outgoing message, the NMM may access the Oracle through a clearly defined interface: make a call to the oracle by producing a query of the form str and expecting (r) with str $= \text{COM}(r)$ in return. Hence, the NMM slowly learns the required random tapes and the associated decommitment information as we proceed in the simulation.

The reason behind such a separation of the simulator into two parts is exactly same as the one in simulation-sound concurrent zero-knowledge (Section E.2).

Furthermore, just like any other MPC simulator, $\mathcal{S}$ will make output queries to the trusted functionality to get the output in each session by providing the input used by the adversary in that session. Looking ahead, such queries will also be made by the machine committed as the non-black box witness. These will be formulated as the oracle queries of the form output$(\cdot)$ defined in the language Λ (Figure 1). Also note that our simulator will already know these outputs which need to be provided in a simulated slot by the time it has to generate the witness for any simulated UA using that slot.

Now we describe our simulator in detail. The NMM handles each outgoing message as described below. In each of these steps, the randomness required by the NMM is taken from the string s_i (assuming that the message will be the i-th message exchanged between the adversary and the NMM). For the sake of simplicity of exposition, below we only describe the case in which the honest party sends the first message in the protocol. The other case, in which the adversary sends the first message, can be handled in an analogous manner and is omitted.

Non-Black-Box Simulation Phase. The simulator simulates the protocol $\langle P, V \rangle_{H \to \mathcal{A}}$ on behalf of the honest party as follows:

1. **The slot begin message z:** The NMM defines a machine Π which simply consists of all the information the NMM has as part of its own state plus the current state of the adversary. However, the machine Π does not have the information that the Oracle stores (which is not yet learnt by the NMM) and also the outputs which have not been computed yet. The NMM then computes $z = \text{COM}(h(\Pi))$ and sends z to the adversary.

2. **A UA message marked blank :** The NMM runs the combinatorial module to see if the UA message is marked blank . If so, the NMM simply generates a random string of appropriate

size and sends a commitment to that string to the adversary.

3. **A UA message marked simulate :** If the combinatorial module marks a UA prover message as simulate , the NMM proceeds as follows. If it is a UA first message, the NMM constructs a witness for the statement $(h, z, r) \in \Lambda$ and uses that to compute the UA first message (details of how the witness is constructed is given later on). The NMM now commits to the resulting UA first message. The resulting UA first message and the opening to the commitment are now stored by the NMM.

   Note that the NMM does not store any other intermediate information resulting from this step. In particular, it discards the witness and other information required to compute the UA first message (such as the PCP). This is done to ensure that the state of the NMM (and hence the machine Π it commits to when a slot begins) is bounded by a fixed polynomial. This is crucial to keep the simulation polynomial time.

   If the message is a last UA prover message such that the corresponding UA first message was also marked simulate , then the NMM *again reconstructs* the witness for the appropriate statement $(h, z, r) \in \Lambda$, reconstructs the UA first message, and then, computes the UA last message depending upon the UA challenge given by the adversary. Now the NMM computes a commitment to this UA last message and sends it to the adversary. The computed message and the opening to the commitment are again stored by the NMM. Note that, as earlier, the NMM discards any intermediate information computed in this step.

   If the message is a UA last message where the UA first message was marked blank , the NMM simply generates a random string of appropriate size and sends a commitment to that string to the adversary.

4. **Non-Malleable Commitment:** The simulator instead of committing to bit 0, sends com_1 as a non-malleable commitment to bit 1 using NMCom and identity of the honest party.

5. **Witness-indistinguishable argument of knowledge:** The NMM uses a witness to the statement: "there exists i such that the i-th UA execution was convincing". By the coverage property of the marking strategy, before we reach the WIAOK stage of a session, there must exist an i such that both the UA first and last messages were marked simulate except with negligible probability (if this doesn't happen, the simulation is aborted). This means that for that i, the NMM must have computed the UA first and last message such that the UA execution becomes convincing. A witness to this statement is simply the openings to the commitments to the UA first and last messages, which is a part of the state of the NMM.

   Observe that, by the properties of the UA system, the computation required in this step is similar for each session and, in particular, does not "grow" as the simulation proceeds (since it depends only on the computational complexity of the UA verifier and not on that of the UA prover).

Constructing the witness. Now we show that for any universal argument which is selected for simulation, there exists a witness for the statement $(h, z, r) \in \Lambda$. This can be seen by the construction of the NMM and the Oracle. We observe the following:

- Consider the point till which the NMM had generated the message z in the transcript. From this point onwards, the NMM, given only queries to the decommitment Oracle and the output

queries to the trusted party, was able to continue generating the transcript and arrive at the challenge r. Thus, the machine Π can perform the same execution since it starts from the same internal state *assuming* it too gets access to the same Oracle query responses and outputs.

- Now consider the point when the first UA message is due which is marked simulate . Let the index of the challenge message r, for the slot picked for simulation, in the transcript be j. The simulator sets $y_1 = j$. Let y_2 contain all the Oracle queries the NMM made while going from the message z to r (both inclusive) and corresponding oracle responses. Furthermore, let y_3 contain all the outputs obtained from the trusted party and the corresponding inputs while going from the message z to r. Now observe that any oracle query Π makes (including both decommit and output queries before it outputs r) can be answered using y_2 and y_3 because its execution would be identical to that of the NMM. Note that since the outputs obtained from the trusted party are valid corresponding to the input used by the adversary, they would be in set S and hence would be accepted by the algorithm $\mathcal{T}$ (see Definition 1).

- Now the witness simply consists of Π, y_1, y_2, y_3, and, the opening of the commitment z. The NMM already has all of these as part of its internal state by the time it reaches r (and in particular, by the time the first UA message is due). Thus, the NMM has computed the witness required to complete the universal argument.

Note that since $\mathcal{S}$ will successfully commit to 1 instead of 0, it will have a valid trapdoor td_1 to be used in input indistinguishable computation phase and trapdoor witness for WI in final secure computation phase.

In the second step of this phase, the adversary runs an instance of $\langle P, V\rangle_{\mathcal{A}\to H}$ preamble and sends com_2 in the NMCOM phase and proves in WIAOK phase that either com_2 is a valid commitment to 0 or one the UA instances in preamble was convincing. Throughout this phase, $\mathcal{S}$ simply uses the honest party strategy to generate all the messages of the honest party except the verifier challenge strings in the preamble. It generates the challenge strings by applying a suitably expanding pseudorandom generator to the randomness given for the message.

Input Indistinguishable Computation Phase. The simulator $\mathcal{S}$ and the adversary $\mathcal{A}$ run an instance of $\langle P_1^{\mathsf{iic}}, P_2^{\mathsf{iic}}\rangle$ for the functionality $f_{\mathsf{com}_1,\mathsf{com}_2}$. $\mathcal{S}$ uses a sufficiently long random string and the trapdoor from the previous phase as input (unlike the honest party that uses its input x_H and randomness r_H and $\perp$ as the trapdoor). Since the trapdoor input by the simulator is valid, it gets the input $x_{\mathcal{A}}$ and the randomness $r_{\mathcal{A}}$ used by the adversary as output of this phase.

Final Secure Computation Phase. Let S_{sh} denote the simulator for the semi-honest two-party protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ used in our construction. $\mathcal{S}$ internally runs the simulator S_{sh} on adversary's input $x_{\mathcal{A}}$. S_{sh} starts executing, and, at some point, it makes a call to the trusted party in the ideal world with some input (say) $x_{\mathcal{A}}$. At this point, $\mathcal{S}$ makes a query $(sid, x_{\mathcal{A}})$ to the trusted functionality $\mathcal{F}$, where *sid* is the session identifier. The output value received from $\mathcal{F}$ is forwarded to S_{sh}. S_{sh} runs further, and finally halts and outputs a transcript $\beta_{H,1}, \beta_{\mathcal{A},1}, \ldots, \beta_{H,q}, \beta_{\mathcal{A},q}$ of the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$, and an associated random string $\hat{r}_{\mathcal{A}}$. $\mathcal{S}$ now performs the following steps.

1. $\mathcal{S}$ first computes the random string $\tilde{r}_{\mathcal{A}}$ such that $\tilde{r}_{\mathcal{A}} = r_{\mathcal{A}} \oplus \hat{r}_{\mathcal{A}}$ and sends $\tilde{r}_{\mathcal{A}}$ to $\mathcal{A}$.

2. Now, in each round $j \in [q]$, $\mathcal{S}$ sends $\beta_{H,j}$. It then engages in an execution of $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle_{H \to \mathcal{A}}$ with $\mathcal{A}$ where $\mathcal{S}$ uses the trapdoor witness (deviating from honest party strategy that used the real witness). Next, on receiving $\mathcal{A}$'s next message $\beta_{\mathcal{A},j}$ in the protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$, $\mathcal{S}$ engages in an execution of $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle_{\mathcal{A} \to H}$ with $\mathcal{A}$ where it uses the honest verifier strategy. Finally at any stage, if the j^{th} message of the adversary is not $\beta_{\mathcal{A},j}$ and the proof $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}} \rangle_{\mathcal{A} \to H}$ given immediately after this messages is accepted, then the simulator aborts all communication and outputs $\perp$. (Later, we establish in the proof of Lemma 6 that $\mathcal{S}$ outputs $\perp$ with only negligible probability.)

This completes the description of our simulator $\mathcal{S}$.

Having defined the simulation, we give some of the extensions of our model. Then in the next section we will show the indistinguishability of the real and simulated executions.

Extensions of our result. Note that the way we have given our result and applications, it already handles the case of interchangeable roles and different parties getting different outputs. Our result can be further extended to more general settings as long as the ideal world experiment satisfies the bounded pseudoentropy condition. Some of these extensions are as follows:

- Adaptive Inputs Setting: Any honest party can choose its input in a session adaptively depending upon its initial state and view in the ideal world so far.
- Multiple Functionalities: Our result would also hold for an ideal world in which different functions are being computed in different sessions as long as the whole ideal world experiment satisfies the bounded pseudoentropy condition.
- Multiparty case: The above protocol can be generalized to the multi-party case using standard ideas. We provide a sketch here and defer the details to the full version. Our protocol Σ is completely symmetric in all steps. In particular, in each stage, first the party P_1 acts and then the party P_2 acts symmetrically. Note that the input commitment phase can be split into two phases: First P_1 uses its inputs and randomness and P_2 uses the trapdoor followed by a similar behavior from the other side. When there are more parties, $(P_3, ...)$, they would follow P_2 and act symmetrically one by one. Note that our proof strategy only uses the fact that the ideal world experiment has bounded pseudoentropy, which refers to the complete output tuple of the adversary.

F.1 Indistinguishability of the Views

We consider two experiments $\mathcal{H}_{\mathsf{Real}}$ and $\mathcal{H}_{\mathsf{Ideal}}$, where $\mathcal{H}_{\mathsf{Real}}$ corresponds to the real execution of Π while $\mathcal{H}_{\mathsf{Ideal}}$ corresponds to the ideal computation of $\mathcal{F}$, as described below. Parts of this section have been taken verbatim from [GGJS11] and [Goy13].

Experiment $\mathcal{H}_{\mathsf{Real}}$: The simulator $\mathcal{S}$ is given the inputs of all the honest parties. By running honest strategy for the honest parties, it generates their outputs along with $\mathcal{A}$'s view. This corresponds to the real execution of the protocol. The output of the hybrid corresponds to the outputs of the honest parties and the view of the adversary $\mathcal{A}$.

Experiment $\mathcal{H}_{\mathsf{Ideal}}$: $\mathcal{S}$ simulates all the sessions without the inputs of the honest parties (in the same manner as explained in the description of $\mathcal{S}$) and outputs the view of $\mathcal{A}$. Each honest party outputs the response it receives from the trusted party. Again the output of the hybrid corresponds to the outputs of the honest parties and the view of the adversary $\mathcal{A}$.

Let ν_i be a random variable that represents the output of $\mathcal{H}_i$. We now claim that the output distributions of $\mathcal{H}_{\mathsf{Real}}$ and $\mathcal{H}_{\mathsf{Ideal}}$ are indistinguishable, as stated below:

Lemma 6 $\nu_{\mathsf{Real}} \stackrel{c}{\equiv} \nu_{\mathsf{Ideal}}$

We will prove this lemma using a carefully designed series of intermediate hybrid experiments. More details are given below.

F.2 Getting Started

We will prove Lemma 6 by contradiction. Suppose that the hybrids $\mathcal{H}_{\mathsf{Real}}$ and $\mathcal{H}_{\mathsf{Ideal}}$ are distinguishable in polynomial time, i.e., there exists a PPT distinguisher D that can distinguish between the two hybrids with a non-negligible probability. We will now consider a series of hybrid experiments $\mathcal{H}_{\mathsf{first}}$ followed by $\mathcal{H}_{i:j}$, where $i \in [n^c]$, and $j \in [7]$ and $\mathcal{H}_{\mathsf{last}}$. For each intermediate hybrid $\mathcal{H}_k$, we define a random variable ν_k that represents the output (including the view of the adversary and the outputs of the honest parties) of $\mathcal{H}_k$.

Below, we show that no PPT distinguisher can distinguish between any two consecutive hybrids, which will establish (via the hybrid arguments) that no polynomial time distinguisher can distinguish between ν_{Real} and ν_{Ideal} with non-negligible probability, which is a contradiction.

In the sequel, we will make use of the notation described in Section F. Also, whenever necessary, we will augment our notation with a super-script that denotes the session number.

Soundness Invariant. Looking ahead, while proving the indistinguishability of the outputs of our hybrid experiments, we will need to argue that in each session $\ell \in [m]$, the commitment $\mathsf{com}_{\mathcal{A}\to H}$ is a commitment to 0 so that $\mathcal{A}$ does not possess a valid trapdoor for input indistinguishable computation phase and the trapdoor condition is false for each instance of $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}}\rangle_{\mathcal{A}\to H}$. We refer to this as the *soundness condition*.

Consider the instance of $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ in session ℓ. Let $\pi^{\ell}_{\mathcal{A}}$ denote the statement proven in $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$. Then, informally speaking, $\pi^{\ell}_{\mathcal{A}}$ states that $\mathcal{A}$ committed to bit 0 (in the NMCOM phase). Note that the soundness condition "holds" if we prove that in each session $\ell \in [m]$, $\mathcal{A}$ commits to a valid witness to the statement $\pi^{\ell}_{\mathcal{A}}$ in the non-malleable commitment $\mathsf{com}_{\mathcal{A}\to H}$ inside $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$. To this end, we define m random variables, $\{\rho^{\ell}_i\}^m_{\ell=1}$, where ρ^{ℓ}_i is the value committed in NMCOM phase in $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ as per ν_i in hybrid $\mathcal{H}_i$.

Now, before we proceed to the description of our hybrids, we first claim that the soundness condition holds in the real execution. We will later argue that the soundness condition still holds as we move from one hybrid to another. This call this soundness invariant.

Lemma 7 *Let $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ and $\pi^{\ell}_{\mathcal{A}}$ be as described above corresponding to the real execution. Then, for each session $\ell \in [m]$, if the honest party does not abort the session (before the first message of the Input Indistinguishable Computation Phase is sent) in the view ν_{Real}, then $\rho^{\ell}_{\mathsf{Real}}$ is a valid witness to the statement $\pi^{\ell}_{\mathcal{A}}$, i.e. $\mathsf{com}_{\mathcal{A}\to H}$ is a valid commitment to 0, except with negligible probability.*

Intuitively, the above lemma follows from the stand alone soundness of $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ (Section E.1) similar to lemma 4.

F.3 Description of the Hybrids

We describe the hybrids $\mathcal{H}_{\mathsf{first}}$, $\mathcal{H}_{i:j}$ for $i \in [n^c]$ and $j \in [7]$ and $\mathcal{H}_{\mathsf{last}}$ below:

Experiment $\mathcal{H}_{\mathsf{first}}$: In this experiment, the simulator is divided into the Oracle and the next message machine (NMM). The Oracle generates the strings $s_1, \ldots, s_{n^c}$, computes the commitments $C = (C_1, \ldots, C_{n^c})$ and gives C to the NMM. The oracle also has the inputs of honest parties for the sessions and computes $c_{x_i} = \text{COM}(x^i_H)$ as commitment of honest party input in i-th session[6]. It gives $C_x = (c_{x_1}, c_{x_2}, \ldots, c_{x_m})$ to NMM. In addition, the NMM generates s required to run the combinatorial module $Mark(\cdot)$. To compute the next message, the NMM queries the Oracle to get the appropriate random string and honest party input for that session (if required) and computes the required message behaving honestly. Furthermore, the NMM runs the combinatorial module (using the random tape s) and internally marks each UA message either simulate or blank . The only change between this hybrid and $\mathcal{H}_{\mathsf{Real}}$ is that the NMM uses a PRG to generate the verifier challenge strings in $\langle P, V\rangle_{\mathcal{A}\to H}$. We claim that

$$\nu_{\mathsf{Real}} \stackrel{c}{\equiv} \nu_{\mathsf{first}} \tag{23}$$

$$\forall \ell \quad \rho^{\ell}_{\mathsf{Real}} \stackrel{c}{\equiv} \rho^{\ell}_{\mathsf{first}} \tag{24}$$

Proving Equation 23. To prove this, we will consider another sequence of hybrids, in which we change the honest verifier challenges one by one from being random to pseudorandom. Let $\mathcal{H}_{\mathsf{Real}:1}$ be $\mathcal{H}_{\mathsf{Real}}$. For $1 \le j \le p$, where p are the total number of honest verifier challenge messages, consider two consecutive hybrids $\mathcal{H}_{\mathsf{Real}:j}$ and $\mathcal{H}_{\mathsf{Real}:(j+1)}$ such that first $(j-1)$ verifier challenges are pseudorandom in both $\mathcal{H}_{\mathsf{Real}:j}$ and $\mathcal{H}_{\mathsf{Real}:(j+1)}$, and the jth verifier challenge in the transcript is random in $\mathcal{H}_{\mathsf{Real}:j}$ and pseudorandom in $\mathcal{H}_{\mathsf{Real}:(j+1)}$. Rest all verifier challenges will be random in both $\mathcal{H}_{\mathsf{Real}:j}$ and $\mathcal{H}_{\mathsf{Real}:(j+1)}$. Now, this j-th verifier challenge will not be generated by the NMM. Instead, it will be given to it externally. Moreover, the NMM will not receive any randomness for this message from the oracle. The indistinguishability of $\nu_{\mathsf{Real}:j}$ and $\nu_{\mathsf{Real}:(j+1)}$ will hold by the security of the PRG. Equation 23 holds by a standard hybrid argument.

Proving Equation 24. Let $\mathcal{H}_{\mathsf{Real}:j}$ and $\mathcal{H}_{\mathsf{Real}:(j+1)}$ be as above. Let us assume there exists a ℓ such that $\rho^{\ell}_{\mathsf{Real}:j}$ and $\rho^{\ell}_{\mathsf{Real}:(j+1)}$ are distinguishable by a PPT distinguisher D. We will create a standalone machine M^* that is identical to $\mathcal{H}_{\mathsf{Real}:j}$, except that it takes jth verifier challenge externally from B and forwards it to $\mathcal{A}$. Additionally, M^* exposes the NMCom inside inside $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ to an external receiver R. If B sends a random string, then (B, M^*, R) system is identical to $\mathcal{H}_{\mathsf{Real}:j}$. On the other hand, if B sends a pseudorandom string, then (B, M^*, R) system is identical to $\mathcal{H}_{\mathsf{Real}:(j+1)}$. Since only one message is changing on the left, by security of PRG and the k-robustness of NMCom, the value committed by M^* to R is indistinguishable in two cases. Equation 2 holds by a standard hybrid argument.

[6]Though the honest party may be playing the role of P_1 in one session and P_2 in some other session, for ease of notation, we will denote the honest party input in i-th session by x_i.

Intuition behind rest of the proof. Next, we shall consider n^c sets of intermediate hybrid experiments $\mathcal{H}_{i:1}, \mathcal{H}_{i:2}, \mathcal{H}_{i:3}, \mathcal{H}_{i:4}, \mathcal{H}_{i:5}, \mathcal{H}_{i:6}, \mathcal{H}_{i:7}$ for $i \in [n^c]$. For each of the hybrids, we will prove that the view of $\mathcal{A}$ in the current hybrid is indistinguishable from the previous hybrid. We will also prove that the soundness condition holds in the current hybrid assuming that it holds in the previous hybrid.

Intuition behind indistinguishability proof. The basic problem we deal with in these hybrid experiments occurs because of non-black-box simulation. We are using some cryptographic primitives in our protocol such as commitment schemes COM and NMCom (providing hiding property), a WIAOK system, an IIC protocol, and an WI protocol. We would go through the hybrids by changing what we commit to (and then rely on the hiding property of the commitment scheme), by changing the witness we use in the WIAOK system (rely on the witness indistinguishability), by changing the input used in the IIC protocol (rely on the input indistinguishability of IIC protocol), and by changing the witness used in WI protocol (again rely on the witness indistinguishability).

Now, say that the NMM starts using the opening of the commitment C_i (to the string s_i) while either preparing the machine it commits to (as part of the slot begin message), or, while completing a simulated universal argument. Then, if a primitive is executed using randomness s_i, we note that its security properties (which we rely on), may no longer hold. For example, if the commitment is prepared using randomness s_i, we can no longer rely on the hiding property of this commitment.

To solve this problem, we shall move forward in the transcript and make changes in a sequential manner. When we are in the i-th set of hybrids, we will maintain the following invariants: (a) the NMM will only use the opening of the commitments C_j for $j < i$, and, (b) from this hybrid onwards, we will only change messages indexed i and higher. We must have already made all the required changes in messages before the i-th message.

Intuition behind soundness invariant proof. To argue the soundness of the current hybrid assuming the soundness condition for the previous hybrid, we will rely on the *k-robustness* of NMCom (Appendix A.6). In each hybrid, we change at most k messages from an honest party to the adversary and expose any non-malleable commitment given by the adversary to an honest external receiver R. Then we argue the indistinguishability of the value committed to by the adversary using k-robustness of NMCom and indistinguishability of those k messages in two hybrids. In a hybrid where we change a non-malleable commitment on the left, we will rely on the non-malleability of NMCom with respect to itself.

Define $\mathcal{H}_{0:7}$ to be the same as $\mathcal{H}_{\mathsf{first}}$.

Experiment $\mathcal{H}_{i:1}$: This experiment is identical to the previous one except in how the message with index i in the transcript is handled by the simulator. In this hybrid, if the i-th message is a slot begin message z, the NMM works as follows. Instead of committing to $h(0)$, the NMM now constructs a machine Π (which consists of the internal state of NMM and that of the adversary), computes the commitment $z = \text{COM}(h(\Pi); s_i)$ using the appropriate randomness s_i and sends it to the adversary.

We now claim that,

$$\nu_{i-1:7} \stackrel{c}{\equiv} \nu_{i:1} \tag{25}$$

$$\forall \ell \quad \rho^{\ell}_{i-1:7} \stackrel{c}{\equiv} \rho^{\ell}_{i:1} \tag{26}$$

Proving Equations 25 *and* 26. We claim that these hold by the (computational) hiding property of the commitment scheme COM and k-robustness of NMCom. The only reason it is non-trivial is that the string C contains a commitment C_i to the randomness s_i required to produce the commitment z. However, we argue that the opening to the commitment C_i is not used by the NMM in any non-trivial way in the interaction with the adversary and hence, the commitment generated by using s_i as randomness is semantically secure. We show this via a sequence of intermediate hybrids.

Experiment $\mathcal{H}_{i:1:1}$: This is identical to $\mathcal{H}_{i:1}$ except that the Oracle, upon receiving a query str, only returns value s.t. $\mathsf{str} = \text{COM}(\mathsf{value}; r)$ if $\mathsf{str} = C_j$ for $j \geq i$. Even given access to such a modified Oracle, the NMM could produce a transcript which is identically distributed. We claim that

$$\nu_{i:1} \overset{s}{\equiv} \nu_{i:1:1} \tag{27}$$

$$\forall \ell \quad \rho^{\ell}_{i:1} \overset{c}{\equiv} \rho^{\ell}_{i:1:1} \tag{28}$$

Proving Equation 27. The openings given by the Oracle to the NMM are used by NMM only to either construct a witness for a simulated UA, or, to commit to a machine Π when a slot begins. However, there is no simulated UA or a commitment to a machine after the message i of the transcript in this experiment. Thus, for a message $j \geq i$, the NMM is not using the opening of the commitment C_j in any way in the experiment. Note that here too the NMM generates the commitment $z = \text{COM}(h(\Pi); s_i)$.

Proving Equation 28. Let us assume that equation 28 is false. That is, $\exists \ell \in [m]$ such that $\rho^{\ell}_{i:1}$ and $\rho^{\ell}_{i:1:1}$ are distinguishable by a probabilistic polynomial time (PPT) distinguisher. In this case, we can create an unbounded adversary that extracts the value contained in the non-malleable commitment and is then able to distinguish between the view of $\mathcal{A}$ in $\mathcal{H}_{i:1}$ and $\mathcal{H}_{i:1:1}$, which is a contradiction by equation 27.

Experiment $\mathcal{H}_{i:1:2}$: This is same as $\mathcal{H}_{i:1:1}$ except that we change the commitment C_i to be $\text{COM}(s'_i)$ such that s'_i is chosen independently. The oracle is identical to the previous one except on query $\mathsf{str} = C_i$, it responds with s_i defined before. We claim that

$$\nu_{i:1:1} \overset{c}{\equiv} \nu_{i:1:2} \tag{29}$$

$$\forall \ell \quad \rho^{\ell}_{i:1:1} \overset{c}{\equiv} \rho^{\ell}_{i:1:2} \tag{30}$$

Proving Equations 29 *and* 30. The first equation holds by the hiding property of COM and the second equation holds by k-robustness of NMCom.

We consider similar experiments $\mathcal{H}_{i-1:7:1}$ and $\mathcal{H}_{i-1:7:2}$ corresponding to $\mathcal{H}_{i-1:7}$ where the NMM generates the commitment $z = \text{COM}(0; s_i)$. We describe them in detail below.

Experiment $\mathcal{H}_{i-1:7:1}$: This is identical to $\mathcal{H}_{i-1:7}$ except that the Oracle, upon receiving a query str, only returns value s.t. $\mathsf{str} = \text{COM}(\mathsf{value}; r)$ if $\mathsf{str} = C_j$ for $j \geq i$. Even given access to such a modified Oracle, the NMM could produce a transcript which is identically distributed to that in $\mathcal{H}_{i-1:7}$. We claim that

$$\nu_{i-1:7} \overset{s}{\equiv} \nu_{i-1:7:1} \tag{31}$$

$$\forall \ell \quad \rho^{\ell}_{i-1:7} \overset{c}{\equiv} \rho^{\ell}_{i-1:7:1} \tag{32}$$

Proving Equations 31 *and* 32. The proof is exactly same as the proof for equations 27 and 28.

Experiment $\mathcal{H}_{i-1:7:2}$: This is same as $\mathcal{H}_{i-1:7:1}$ except that we change the commitment C_i to be $\text{COM}(s_i')$ such that s_i' is chosen independently. The oracle is identical to the previous one except on query $\mathsf{str} = C_i$, it responds with s_i defined before. We claim that

$$\nu_{i-1:7:1} \stackrel{c}{\equiv} \nu_{i-1:7:2} \tag{33}$$

$$\forall \ell \quad \rho^{\ell}_{i-1:7:1} \stackrel{c}{\equiv} \rho^{\ell}_{i-1:7:2} \tag{34}$$

Proving Equations 33 *and* 34. The proof is exactly same as the proof for equations 29 and 30.

Note that to prove equations 25 and 26 it is sufficient to prove the following:

$$\nu_{i-1:7:2} \stackrel{c}{\equiv} \nu_{i:1:2} \tag{35}$$

$$\forall \ell \quad \rho^{\ell}_{i-1:7:2} \stackrel{c}{\equiv} \rho^{\ell}_{i:1:2} \tag{36}$$

Proving Equation 35. Note that in experiments $\mathcal{H}_{i-1:7:2}$ and $\mathcal{H}_{i:1:2}$, no information is revealed about the string s_i, which is used as randomness to generate the commitment z. Now NMM instead of generating the message z itself by querying the oracle for C_i, takes the message z externally under randomness s_i and does not make any query for C_i. If it receives a commitment to $h(0)$ then the experiment is identical to $\mathcal{H}_{i-1:7:2}$. If it receives a commitment to $h(\Pi)$, then the experiment is identical to $\mathcal{H}_{i:1:2}$. Since s_i is semantically secure and hence any commitment using s_i as randomness is also semantically secure. Thus, the experiments $\mathcal{H}_{i-1:7:2}$ and $\mathcal{H}_{i:1:2}$ are computationally close. Hence, the view of the adversary in experiments $\mathcal{H}_{i-1:7}$ and $\mathcal{H}_{i:1}$ is computationally close.

Proving Equation 36. It holds by the semantic security of commitment z and the k-robustness of NMCom. More precisely, let us first assume that the claim is false, i.e., $\exists \ell \in [m]$ such that $\rho^{\ell}_{i-1:7:2}$ and $\rho^{\ell}_{i:1:2}$ are distinguishable by a PPT distinguisher D. We will create a standalone machine M^* that is identical to $\mathcal{H}_{i-1:7:2}$, except that instead of simply committing to $h(0)$ using COM, M^* takes this commitment from an external sender B and "forwards" it internally to $\mathcal{A}$. Additionally, M^* "exposes" the NMCom in $\langle P, V \rangle^{\ell}_{\mathcal{A} \to H}$ to an external receiver R. This is feasible because we chose η to be more than communication complexity of receiver in NMCom plus the length of commitment z. Hence, both the receiver messages in NMCom and commitment z can be given externally to NMM.

Let us describe the interaction between M^* and B in more detail. M^* first sends the program Π to B. Now, when B sends the message z, M^* forwards this message from B to $\mathcal{A}$. Note that if B commits to $h(0)$ in the COM execution, then the (B, M^*, R) system is identical to $\mathcal{H}_{i-1:7:2}$. On the other hand, if B commits to the program $h(\Pi)$, then the (B, M^*, R) system is equivalent to $\mathcal{H}_{i:1:2}$. From the k-robustness of NMCom and hiding property of COM, we establish that the value committed by M^* to R must be computationally indistinguishable in both cases.

Experiment $\mathcal{H}_{i:2}$: This experiment is identical to the previous one except how the message with index i in the transcript is handled by the simulator. In this hybrid, if the i-th message is a UA message which is marked $\mathsf{simulate}$, the way the NMM handles this message is identical to the final NMM (described earlier in Section F). In particular, the NMM starts using the witness guaranteed by the construction of Π for computing this UA message (as opposed to just committing to random strings). The intermediate information resulting from this step is discarded (as described earlier).

We now claim that,

$$\nu_{i:1} \stackrel{c}{\equiv} \nu_{i:2} \tag{37}$$

$$\forall \ell \quad \rho^{\ell}_{i:1} \stackrel{c}{\equiv} \rho^{\ell}_{i:2} \tag{38}$$

Proving Equation 37. This relies on the computational hiding property of the commitment scheme COM using which the UA message is committed and is very similar to the previous indistinguishability proof. The difference between this hybrid and the previous one is only in the computation of the i-th message (in case it is a UA prover message marked as simulate). However, similar to the previous indistinguishability argument, we can argue that the randomness used to compute the commitment to this UA prover message is not used by the NMM in anyway in the interaction with the adversary.

Proving Equation 38. After we prove that the randomness used to compute this commitment is not used by NMM anywhere, this equation follows by the hiding property of COM and k-robustness of NMCom.

Experiment $\mathcal{H}_{i:3}$: This experiment in identical to the previous one except that now in all the sessions in which the preamble phase of $\langle P, V\rangle_{H\rightarrow\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript, the NMM starts using the alternative witness (the opening to the simulated UA messages) to complete the WIAOK. For any such session, up to (and including) the i-th message of the transcript, there exists a UA both of whose prover messages are marked simulate . The NMM now starts using the openings of these UA messages as the witness to complete the WIAOK execution. Note that there can be at most one session, in which the preamble phase of $\langle P, V\rangle_{H\rightarrow\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript and not completed upto (and including) the $(i-1)$th message of the transcript. That is there is only session where we are changing the witness in WIAOK compared to the previous hybrid. We now claim that,

$$\nu_{i:2} \stackrel{c}{\equiv} \nu_{i:3} \tag{39}$$

$$\forall \ell \quad \rho^{\ell}_{i:2} \stackrel{c}{\equiv} \rho^{\ell}_{i:3} \tag{40}$$

Proving Equation 39. This is very similar to the previous proof of indistinguishability. The only changes made in this hybrid occur after the message with index i in the transcript. However, as argued earlier, we can prove that the random strings used to compute the prover messages in WIAOK are still semantically secure. Then we can use an external prover to generate the messages of this WIAOK and give these as input to NMM. This can be done since η is chosen to be more than the length of prover messages in WIAOK. The claim follows by the witness indistinguishability of the WIAOK protocol.

Proving Equation 40. This is very similar to the proof of soundness condition in previous hybrids. Once we have proved that the randomness used to generate these prover messages of WIAOK are semantically secure, this equation follows by witness indistinguishability of the WIAOK protocol and k-robustness of NMCom. Since η is chosen to be greater than the communication complexity of prover messages of WIAOK and receiver messages of NMCom, we can take the prover messages of WIAOK externally and expose any of the commitments under NMCom given by the adversary to an external receiver R as above. Also, since k is chosen to be greater than 3, which is the round

complexity of WIAOK, the indistinguishability follows by WI property of WIAOK and k-robustness of NMCom.

Experiment $\mathcal{H}_{i:4}$: This experiment is identical to the previous one except that now in all the sessions in which the preamble phase of $\langle P, V\rangle_{H\to\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript, the NMM generates $\mathsf{com}_{H\to\mathcal{A}}$ as a commitment to 1 instead of 0 in the NMCOM phase. Note that, as argued above, there is only session where we are changing the value committed by NMCom compared to the previous hybrid. We now claim that,

$$\nu_{i:3} \stackrel{c}{\equiv} \nu_{i:4} \tag{41}$$

$$\forall \ell \quad \rho^{\ell}_{i:3} \stackrel{c}{\equiv} \rho^{\ell}_{i:4} \tag{42}$$

Proving Equation 41. This is very similar to the previous proof of indistinguishability. The only changes made in this hybrid occur after the message with index i in the interaction transcript. However, as argued earlier, we can prove that the random strings used to compute this commitment are still semantically secure. Next, we can provide this commitment externally to the NMM since we set η more than the sender communication complexity of NMCom. The claim follows by the hiding property of NMCom.

Proving Equation 42. Once we have proved that the randomness used to generate these sender messages of NMCom are semantically secure, this equation follows by hiding property of the NMCom and non-malleability of NMCom w.r.t. itself. More precisely, let us assume that $\exists \ell \in [m]$ such that $\rho^{\ell}_{i:3}$ and $\rho^{\ell}_{i:4}$ are distinguishable by a PPT distinguisher D. We will create a standalone machine M^* that is identical to $\mathcal{H}_{i:3}$, except that instead of simply committing to witness using NMCom in $\langle P, V\rangle_{H\to\mathcal{A}}$, M^* takes this commitment from an external sender B and "forwards" it internally to $\mathcal{A}$. Additionally, M^* "exposes" the NMCom in $\langle P, V\rangle^{\ell}_{\mathcal{A}\to H}$ to an external receiver R. This can be done since η is chosen to be greater than the communication complexity of sender messages of NMCom plus the communication complexity of receiver messages of NMCom. Let us describe the interaction between M^* and B in more detail. When B starts the execution of NMCom in $\langle P, V\rangle_{H\to\mathcal{A}}$, M^* forwards the messages from B to $\mathcal{A}$; the responses from $\mathcal{A}$ are forwarded externally to B. Note that if B commits to the 0, then the (B, M^*, R) system is identical to $\mathcal{H}_{i:3}$. On the other hand, if B commits to 1, then the (B, M^*, R) system is equivalent to $\mathcal{H}_{i:4}$. From the non-malleability property of NMCom, we establish that the value committed by M^* to R must be computationally indistinguishable in both cases.

Experiment $\mathcal{H}_{i:5}$: This experiment is identical to the previous one except that now in all the sessions in which the preamble phase of $\langle P, V\rangle_{H\to\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript, the NMM uses the trapdoor witness (instead of the real witness) in each instance of WI where the honest party plays the role of the prover in the final secure computation phase. Note that the trapdoor witness for each of these WI must be available to NMM at this point since in the previous hybrid it committed to bit 1. Also note that there can be at most one session, in which the preamble phase of $\langle P, V\rangle_{H\to\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript and not completed upto (and including) the $(i-1)$th message of the transcript. That is there is only session where we are changing the witnesses in WI compared to the previous

hybrid. We now claim that

$$\nu_{i:4} \stackrel{c}{\equiv} \nu_{i:5} \tag{43}$$

$$\forall \ell \quad \rho^{\ell}_{i:4} \stackrel{c}{\equiv} \rho^{\ell}_{i:5} \tag{44}$$

Proving Equation 43. To prove this we will consider another sequence of hybrids, in which we change the witness used in WI phase of that session one by one. Let $\mathcal{H}_{i:4:1}$ be $\mathcal{H}_{i:4}$. For $1 \leq j \leq q$, where q are the number of WI executions in one session, consider two consecutive hybrids $\mathcal{H}_{i:4:j}$ and $\mathcal{H}_{i:4:(j+1)}$ such that first $(j-1)$ instances of WI use trapdoor witnesses in both $\mathcal{H}_{i:4:j}$ and $\mathcal{H}_{i:4:(j+1)}$, and the jth instances uses real witness in $\mathcal{H}_{i:4:j}$ and trapdoor witness in $\mathcal{H}_{i:4:(j+1)}$. Rest all WI instances use real witness both $\mathcal{H}_{i:4:j}$ and $\mathcal{H}_{i:4:(j+1)}$.

As argued earlier, we can prove that the random strings used to compute the prover messages in jth instance of WI are still semantically secure. Then we can use an external prover to generate the messages of this WI and give these as input to NMM, since θ is chosen to be more than the length of prover messages in WI. The indistinguishability of $\nu_{i:4:j}$ and $\nu_{i:4:(j+1)}$ follows by the witness indistinguishability of the WI. Then by a standard hybrid argument, we can extend this proof for equation 43.

Proving Equation 44. Let $\mathcal{H}_{i:4:j}$ and $\mathcal{H}_{i:4:(j+1)}$ be as above. Then we claim that

$$\forall \ell \quad \rho^{\ell}_{i:4:j} \stackrel{c}{\equiv} \rho^{\ell}_{i:4:j+1}$$

Let us first assume that the claim is false, i.e., $\exists \ell \in [m]$ such that $\rho^{\ell}_{i:4:j}$ and $\rho^{\ell}_{i:4:j+1}$ are distinguishable by a PPT distinguisher D. We will create a standalone machine M^* that is identical to $\mathcal{H}_{i:4:j}$, except that instead of behaving honestly in WI, M^* takes this proof from an external prover B and "forwards" it internally to $\mathcal{A}$. Additionally, M^* "exposes" the NMCom in $\langle P, V \rangle^{\ell}_{\mathcal{A} \to H}$ to an external receiver R as described earlier. Let us describe the interaction between M^* and B in more detail. M^* first sends the real and the trapdoor witnesses to B. Now, when B starts the execution of WI, M^* forwards the messages from B to $\mathcal{A}$; the responses from $\mathcal{A}$ are forwarded externally to B. Note that if B uses the real witness, then the (B, M^*, R) system is identical to $\mathcal{H}_{i:4:j}$. On the other hand, if B uses the trapdoor witness, then the (B, M^*, R) system is equivalent to $\mathcal{H}_{i:4:j+1}$. Since k is chosen to be more than the round complexity of WI, the claim follows by k-robustness of NMCom.

Experiment $\mathcal{H}_{i:6}$: This experiment in identical to the previous one except that now in all the sessions in which the preamble phase of $\langle P, V \rangle_{H \to \mathcal{A}}$ has completed up to (and including) the i-th message of the transcript, the NMM uses a sufficiently long random string and the trapdoor from the previous phase as input (unlike the honest party that uses its input x_H and randomness r_H and $\perp$ as the trapdoor). Since the trapdoor input by the simulator is valid, it gets the input $x_{\mathcal{A}}$ and the randomness $r_{\mathcal{A}}$ used by the adversary in that session as output of this phase. We now claim that,

$$\nu_{i:5} \stackrel{c}{\equiv} \nu_{i:6} \tag{45}$$

$$\forall \ell \quad \rho^{\ell}_{i:5} \stackrel{c}{\equiv} \rho^{\ell}_{i:6} \tag{46}$$

Proving Equation 45. This is very similar to the previous proof of indistinguishability. As argued earlier, we can prove that the random strings used to compute the honest party messages in IIC

are still semantically secure. Then we can use an external party to generate the messages of IIC and give these as input to NMM, since θ is chosen to be more than the length of messages of any party in IIC. We have already established from the previous hybrids that the *soundness condition* holds at this point. This means that $\mathcal{A}$ does not possess a valid trapdoor for this phase. Hence, his output and view are indistinguishable in the two hybrids by the input indistinguishability of the IIC.

Proving Equation 46. This is very similar to the proof of equation 40. Once we have proved that the randomness used to generate honest party's messages in IIC are semantically secure, this equation follows by input indistinguishability of the IIC protocol and k-robustness of NMCom. Since θ is chosen to be greater than the communication complexity of any party in IIC and receiver messages of NMCom, we can take the party's messages of IIC externally and expose any of the commitments under NMCom given by the adversary to an external receiver R as above. Also, since k is chosen to be greater than the round complexity of IIC, the indistinguishability of committed values by $\mathcal{A}$ follows by k-robustness of NMCom.

Experiment $\mathcal{H}_{i:7}$: This experiment in identical to the previous one except that now in all the sessions in which the preamble phase of $\langle P, V\rangle_{H\to\mathcal{A}}$ has completed up to (and including) the i-th message of the transcript, the NMM simulates the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ as follows: Let $s(i)$ be the session number for the session whose preamble phase ended at the i-th message. Then compared to the previous hybrids, we are changing the $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ in session $s(i)$. Let S_{sh} denote the simulator for the semi-honest two-party protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ used in our construction. NMM internally runs the simulator S_{sh} on adversary's input $x_{\mathcal{A}}$. S_{sh} starts executing, and, at some point, it makes a call to the trusted party in the ideal world with some input (say) $x_{\mathcal{A}}$. At this point, NMM makes a query $(s(i), x_{\mathcal{A}})$ to the trusted functionality $\mathcal{F}$. The output value received from $\mathcal{F}$ is forwarded to S_{sh}. S_{sh} runs further, and finally halts and outputs a transcript $\beta_{H,1}, \beta_{\mathcal{A},1}, \ldots, \beta_{H,q}, \beta_{\mathcal{A},t}$ of the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$, and an associated random string $\hat{r}_{\mathcal{A}}$.

Now, NMM forces this transcript and randomness on $\mathcal{A}$ in the same manner as described in section F. We claim that during the execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$, each reply of $\mathcal{A}$ must be consistent with this transcript, except with negligible probability. Note that we have already established from the previous hybrids that the *soundness condition* holds (except with negligible probability) at this point. This means that the trapdoor condition is false for each instance of $\langle P_{\mathsf{wi}}, V_{\mathsf{wi}}\rangle^{s(i)}_{\mathcal{A}\to H}$. By the soundness property of WI used in our construction, each reply of $\mathcal{A}$ must be consistent with this transcript. We now claim that

$$\nu_{i:6} \stackrel{c}{\equiv} \nu_{i:7} \tag{47}$$

$$\forall \ell \quad \rho^{\ell}_{i:6} \stackrel{c}{\equiv} \rho^{\ell}_{i:7} \tag{48}$$

Proving Equation 47. Informally speaking,this follows from the semi-honest security of the two-party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}}\rangle$ used in our construction. We now give more details.

Now our NMM will get the protocol transcript of semi-honest 2PC externally from an external sender B and forces it on $\mathcal{A}$ instead of generating it using S_{sh} as follows: NMM first queries the ideal world trusted party on the extracted input of $\mathcal{A}$ for session $s(i)$ in the same manner as explained above. Let $x_{\mathcal{A}}^{s(i)}$ denote the extracted input of $\mathcal{A}$. Let $x_H^{s(i)}$ denote the input of the honest party in session $s(i)$. Let O be the output that NMM receives from the trusted party. Now NMM sends $x_H^{s(i)}$ (obtained from the oracle) along with $x_{\mathcal{A}}^{s(i)}$ and O to B and receives from B a transcript for

$\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ and an associated random string. NMM forces this transcript and randomness on $\mathcal{A}$ in the same manner as $\mathcal{S}$ does. Now, the following two cases are possible:

1. B computed the transcript and randomness by using *both* the inputs - $x_H^{s(i)}$ and $x_{\mathcal{A}}^{s(i)}$ - along with the output O. In this case, the transcript output by B is a real transcript of an honest execution of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$.

2. B computed the transcript and randomness by using only adversary's input $x_{\mathcal{A}}^{s(i)}$, and the output O. In this case B simply ran the simulator S_{sh} on input $x_{\mathcal{A}}^{s(i)}$ and answered its query with O. The transcript output by B in this case is a simulated transcript for $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$.

In the first case, the (B, M) system is identical to $\mathcal{H}_{i:6}$, while in the second case, the (B, M) system is identical to $\mathcal{H}_{i:7}$. By the (semi-honest) security of $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$, we establish that the view of $\mathcal{A}$ must be indistinguishable in both the cases, except with negligible probability.

Proving Equation 48. We will leverage the semi-honest security of the two-party computation protocol $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ and the k-robustness of the non-malleable commitment scheme NMCom to prove equation 48.

Specifically, we will construct a standalone machine M^* that is identical to M as described above, except that it "exposes" the NMCom in $\langle P, V \rangle^{\ell}_{\mathcal{A} \to H}$ to an external receiver R, as described earlier. Note that if B produces a transcript $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ according to case 1 (as described above), then the (B, M^*, R) system is identical to $\mathcal{H}_{i:6}$. On the other hand, if B produces a transcript for $\langle P_1^{\mathsf{sh}}, P_2^{\mathsf{sh}} \rangle$ according to case 2, then the (B, M^*, R) system is identical to $\mathcal{H}_{i:7}$. However, since this interaction with B consists of a single message, which is computationally indistinguishable in the two cases, the commitment of M^* to R is computationally indistinguishable by k-robustness of NMCom.

This way we create the hybrids up to $\mathcal{H}_{n^c:7}$. Note that in this last hybrid, NMM does not use the honest party inputs to generate any message. Hence, it does not query the oracle with the decommitment for any c_{x_i}. So in the final hybrid, we remove the honest party inputs from the oracle. Intuitively, this is possible since oracle is the external uncommitted part of the simulator.

Experiment $\mathcal{H}_{\mathsf{last}}$: This hybrid is same as $\mathcal{H}_{n^c:7}$ except that now the oracle generates c_{x_i} differently. For each i, it picks a sufficiently long string r_i and computes $c_{x_i} = \text{COM}(r_i)$ and gives these commitments to NMM. We claim that

$$\nu_{n^c:7} \stackrel{c}{\equiv} \nu_{\mathsf{last}} \tag{49}$$

$$\forall \ell \quad \rho^{\ell}_{n^c:7} \stackrel{c}{\equiv} \rho^{\ell}_{\mathsf{last}} \tag{50}$$

Proving Equation 49. Since NMM in $\mathcal{H}_{n^c:7}$ never queries the oracle for decommitments of any c_{x_i}, the transcript in this hybrid is computationally indistinguishable from the previous hybrid by security of COM.

Proving Equation 50. To prove this we can consider a sequence of intermediate hybrids where we change the commitments c_{x_i} one by one. The computational indistinguishability of ρ^{ℓ} between any

two such consecutive hybrids will follow by the hiding property of COM and k-robustness of NMCom.

Note that $\mathcal{H}_{\mathsf{last}}$ is same as $\mathcal{H}_{\mathsf{Ideal}}$. Hence, we have proved that $\nu_{\mathsf{Real}} \stackrel{c}{\equiv} \nu_{\mathsf{Ideal}}$.

G Related Work on Concurrent Computation in Plain Model

As mentioned in introduction, a rewinding based simulator for concurrent computation gets into the problem of multiple output queries to the ideal functionality. Trying to solve this bottleneck of "handling extra queries" in various ways has inspired a number of different works. Below we describe these relaxed models.

- **Resettable and stateless computation:** In the resettable setting, Goyal and Sahai [GS09] (and more recently Goyal and Maji [GM11]) were able to obtain positive results by resetting the ideal functionality to obtain multiple outputs per session.
- **Multiple ideal query model:** Goyal, Jain and Ostrovsky [GJO10] proposed the multiple ideal-query model, where for every session in the real world, the simulator is allowed to query the functionality multiple times in the ideal world. Subsequent works [GJ13, GGJ13] used these ideas to resolve seemingly unrelated open problems such as concurrent precise zero-knowledge [PPS+08], concurrent password-based key exchange [GL01], bounded concurrent secure computation with graceful degradation [Lin03a], etc.
- **The plain model:** In the plain model, recently Goyal [Goy12] was able to obtain positive results for a large class of functionalities in the concurrent single input setting by constructing an "output predictor" to be utilized by the simulator during rewinds to answer the additional outputs.
- **Leaky ideal query model and others:** Recently, Goyal, Gupta and Jain [GGJ13] proposed the *leaky ideal query model* where the simulator is allowed to leak upon the secret state of the trusted functionality in the ideal world. This "extra" information allowed them to answer the extra output queries. This paradigm has also been relevant to constructing protocols in the super-polynomial simulation and input-indistinguishable computation models [GGJS12].