Refuting the Dream XOR Lemma via Ideal Obfuscation and Resettable MPC

Saikrishna Badrinarayanan [*] Yuval Ishai [†] Dakshita Khurana [‡] Amit Sahai [§]
Daniel Wichs [¶]

Abstract

We provide counterexamples to the "dream" version of Yao's XOR Lemma. In particular, we put forward explicit candidates for hard predicates, such that the advantage of predicting the XOR of many independent copies does not decrease beyond some fixed negligible function, even as the number of copies gets arbitrarily large.

We provide two such constructions:

- Our first construction is in the ideal obfuscation model (alternatively, assuming virtual black-box obfuscation for a concrete class of circuits). It develops a general framework that may be of broader interest, and allows us to embed an instance of a *resettably*-secure multiparty computation protocol into a one-way function. Along the way, we design the first resettably-secure multiparty computation protocol for general functionalities in the plain model with super-polynomial simulation, under standard assumptions.
- The second construction relies on public-coin differing-inputs obfuscation (PCdiO) along with a certain form of hash-function security called extended second-preimage resistance (ESPR). It starts with a previously known counterexample to the dream direct-product hardness amplification based on ESPR, and uses PCdiO to upgrade it into a counterexample for the XOR lemma.

Prior to our work, even completely heuristic counterexamples of this type were not known.

[*]Snap. Email: bsaikrishna7393@gmail.com. Part of the work done while at UCLA.

[†]Technion. Email: yuval.ishai@gmail.com. Partially supported by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20.

[‡]University of Illinois, Urbana-Champaign. Email: dakshkhurana@gmail.com. Supported in part from a Visa Research Faculty Award, DARPA SIEVE, NSF QIS and an award from C3AI.

[§]UCLA and Center for Encrypted Functionalities. Email: amitsahai@gmail.com. Supported in part from a Simons Investigator Award, DARPA SIEVE award, NTT Research, NSF Frontier Award 1413955, BSF grant 2018393, a Xerox Faculty Research Award, a Google Faculty Research Award, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through Award HR00112020024.

[¶]Northeastern University and NTT Research. Email: danwichs@gmail.com. Daniel Wichs is partially supported by NSF grants CNS-1750795, CNS- 2055510 and the Alfred P. Sloan Research Fellowship.

Contents

1 Introduction

Consider the following standard information-theoretic technique for hardness amplification. Suppose we have a joint distribution (X, B) such that the bit B is "weakly unpredictable" given X, in the sense that B has positive entropy conditioned on X. Then, for t independent samples (x_i, b_i) from (X, B), the XOR of all b_i can only be predicted from $(x_1, \ldots, x_t)$ with $2^{-\Omega(t)}$ advantage over a random guess. The question we address in this work is whether the same holds also in the computational setting.

The computational variant of the above XOR lemma is a central tool in complexity theory that first appeared in presentations of Yao's work [Yao82]. It postulates that if a predicate P of an input $x \in \{0,1\}^\lambda$ is weakly unpredictable by algorithms of a certain complexity, with respect to some input distribution $X = X(\lambda)$, then $P(x_1, \ldots, x_t) = \bigoplus_{i=1}^t P(x_i)$ for large enough t is strongly unpredictable by algorithms within a related complexity bound. Yao stated this in the context of one-way functions, where the predicate P can be any "hard-core" bit of a one-way function f: in other words, P is an easy-to-compute Boolean function of the input to f that is hard to predict given the output of f. Here the distribution X is the output distribution of f on a uniformly random input.

Since it was first introduced, different versions of this lemma were proved in the literature, starting with Levin's proof [Lev85], an alternate proof by Impagliazzo [Imp95], and a third one by Goldreich et al. [GNW11]. Unfortunately, all existing proofs of the XOR lemma are stuck at the following barrier: for *any fixed* negligible function $\mu(\cdot)$, no matter how large the polynomial $t(\cdot)$ is, we cannot prove that for large enough λ, the adversary's advantage drops to $\mu(\lambda)$. More concretely, we have no evidence that *any* polynomial $t(\lambda)$ number of repetitions bring the adversary's advantage down even to $\lambda^{-\log \lambda}$, if the original hardness of f was assumed to hold only against all polynomial-sized adversaries.

It is unclear why this barrier exists, and whether it is just an artifact of known proof techniques. Intuitively, it appears that the adversary's advantage should reduce arbitrarily if we perform sufficiently many repetitions. This was previously conjectured and termed the "dream version" of Yao's XOR lemma. It was formalized in [GNW11], and used by [BGI08] to obtain (a stronger flavor of) weak public-key cryptography from strong one-way functions. The dream XOR lemma, if true, would fundamentally change our understanding of how intractability works. It would help arbitrarily bring down errors in security arguments with sufficiently many repetitions; leading to exciting new constructions of primitives like non-interactive non-malleable commitments following [BL18], or easily obtain multi-instance security [BRT12] with good parameters from standard hardness assumptions.

Previous Explanations. Over the years, there have been some explanations for why the dream XOR lemma eludes a proof. Black-box reduction based proofs of the dream XOR lemma are likely to fail for the following folklore reason. In order to prove that XOR parallel repetition brings the adversary's advantage down to some small probability $\mu(\lambda)$, we would need an efficient reduction that uses an adversary breaking security of the parallel repetition with advantage $\mu(\lambda)$, to break the security of a single instance with significantly larger advantage. But such a reduction cannot obtain any useful information from an attacker unless it succeeds at least once, and this could require $\mathsf{poly}(1/\mu(\lambda))$ attempts. Therefore, this reduction could be efficient for every inverse polynomial $\mu(\cdot)$, but would become inefficient as soon as $\mu(\cdot)$ is a fixed negligible function. This has been attributed to Rudich [GNW11], who also proved that the dream XOR lemma does not hold in a relativized world, by introducing an oracle that inverts every tuple of instances with probability $\mu(\lambda)$.

Shaltiel and Viola [SV10] initiated a line of work on the impossibility of better black-box hardness amplification results, including the XOR lemma. A recent work by Shaltiel [Sha20], improving on [GSV18], rules out dream XOR lemmas with proofs by so-called "class reductions," which can exploit the efficiency of their oracle. Despite this progress, it is not clear that ruling out (even relaxed forms of) black-box reductions gives a strong evidence against the dream XOR lemma. There are quite a few examples for the surprising power of non-black-box techniques, both in cryptography and in complexity theory.

Another partial explanation was offered by Dodis et al. [DJMW12], who gave a counterexample to the "dream version" of the "direct-product lemma". In particular, they showed how to construct a *weak one-way function* that no polynomial-time adversary can invert with probability better than (e.g.,) $\frac{1}{2}$, but any arbitrary polynomial number of independent copies can be simultaneously inverted in polynomial time with advantage greater than $\lambda^{-\log\lambda}$. Their construction relies on an ad-hoc assumption on an un-keyed hash function, which was justified in the random-oracle model with auxiliary input. Since the direct-product lemma implies the XOR lemma without much loss in parameters, a dream version of the former would have implied the latter. Therefore, their work closes off one potential avenue toward proving the dream XOR lemma. Had their weak one-way function also been injective, then their counter-example to the dream direct-product lemma would have also immediately yielded a counter-example to the XOR lemma by taking the hard-core bits of the function. However, their weak one-way function was not injective, in which case the ability to find *some* pre-images and break one-wayness, does not imply the ability to find *the* correct hard-core bits and break the XOR lemma.

In this work, we ask the following question, which was explicitly left as an open problem in [DJMW12]:

Can we build an explicit counterexample to the dream version of the XOR lemma: a one-way function with a (weakly) hardcore bit, for which predicting the XOR of many hardcore bits does not reduce an adversary's advantage beyond a specific negligible function?

We note that, using a trusted setup that generates a trapdoor permutation (and can therefore invert it), one could heuristically obfuscate Rudich's oracle to obtain an explicit counterexample in the structured reference string model; however, beyond the need for trusted setup, this would induce a strong correlation between different instances. Such a correlation is inherently at odds with the idea of the dream XOR lemma, which conjectures hardness of *completely independent and uncorrelated instances.*

Are There Simple Heuristic Counterexamples? To our knowledge, prior to our work, there were no candidate counterexamples to the dream XOR lemma *under any assumption.* In other words, there was not even a *heuristic* construction of an explicit predicate that can be plausibly conjectured to violate the dream XOR lemma. The difficulty of coming up with such heuristic counterexamples is arguably why the conjecture was put forward in the first place.

1.1 Overview of Results

We give two kinds of explicit counterexamples to the dream XOR lemma.

Our first counterexample develops a general framework that allows us to embed an instance of multiparty computation with *resettable* security into a non-interactive cryptosystem, such as a one-way function. It relies on ideal obfuscation, or alternatively, virtual-black-box (VBB) obfuscation for some specific class of circuits, along with other standard hardness assumptions. Using this framework, the counterexample to the dream XOR lemma is extremely simple. We believe this framework may be of broader interest and may potentially be useful in designing counterexamples to other conjectures in cryptography. Along the way, we also develop the first resettably-secure multiparty computation protocol for general functionalities in the plain model with super-polynomial simulation, under standard assumptions. This protocol requires three rounds and assumes the existence of two-round sub-exponentially secure statistically sender-private OT. This result is of independent interest and achieves general feasibility for resettably-secure multiparty computation in the plain model; we evade prior impossibility results [GM11] by aiming for super-polynomial simulation.

Our second counterexample is a tailor-made construction whose sole purpose is to defy the dream XOR lemma. However, it avoids the need for ideal or VBB obfuscation. Instead, we can rely on a concrete obfuscation security property called *public-coins differing inputs obfuscation (PCdiO)* [BGI$^+$01, ABG$^+$13, BCP14, IPS15], along with a hash function security property called *extended second-preimage resistance* (ESPR) [DJMW12] and injective one-way functions. The construction uses PCdiO to "upgrade" the counterexample of [DJMW12] for the dream direct-product lemma based on ESPR, into a counterexample for the dream XOR lemma. PcdiO is a clean assumption which, as of today, all existing iO constructions can be conjectured to satisfy. While diO (without the public coin restriction) is known to be implausible [GGHW14], all known implausibility results crucially rely on "contrived" auxiliary information. All such negative results therefore do not generalize to PcdiO (we refer the reader to [IPS15] for further discussion of PcdiO). Indeed, given recent progress on constructing iO (e.g. [JLS21]), it is plausible that the PcdiO, that we rely on, may be reduced to well-studied assumptions.

We stress that even with ideal obfuscation, Rudich's oracle does not directly give rise to such counterexamples because it is not efficient. We believe that our techniques for obfuscation-based counterexamples will find other applications.

Are the Counterexamples Explicit? Both of our counterexamples can be made fully explicit by making the same kind of leap of faith one makes when instantiating standard idealized models in cryptography (such as the random oracle model [BR93] or the generic group model [Sho97]). For the first counterexample, one can use any existing iO construction (such as the one from [JLS21]) as a heuristic substitute for special-purpose obfuscation. A similar heuristic has been suggested in prior works (see, e.g., [GGHW14, GHRW14]). For the second, use iO construction as the PC-diO and use SHA-3 as the ESPR. Either way, one gets fully explicit constructions of one-way functions and hard-core bits for which the XOR lemma *provably* does not amplify hardness. Assuming that the function is actually one-way to begin with relies on a strong but explicit assumption. Nevertheless, if one wanted to prove that the dream XOR lemma holds, one would now have to show an attack against the one-wayness of these explicit one-way function candidates. This is very different from the previous oracle-based separations or (generalized) black-box impossibility results, which could be potentially circumvented by finding a novel non-black-box proof technique. We refer the reader to the end of Section 1.2 for additional discussions about the special-purpose obfuscation assumption.

1.2 First Counterexample

A New Paradigm. We suggest a new paradigm for obtaining counterexamples to parallel repetition. We use this paradigm to obtain an explicit counterexample to the dream version of the XOR lemma.

Our paradigm can be thought of as implementing Rudich's oracle in a distributed manner between completely independent instances of a one-way function. Such distributed protocols are often cryptographically achievable via secure multi-party computation (MPC). Specifically, one could treat each instance of a one-way function as a participant in an MPC protocol, and implement an ideal functionality that with probability $\mu(\lambda)$, outputs the inverse of all the instances of the one-way function. Clearly, this would allow the XOR of the hardcore bits of individual instances of the one-way function to be predicted with an advantage of at least $\mu(\lambda)$. Indeed, a similar idea was employed by [DJMW12] in the context of a counterexample for the direct-product hardness amplification of signature schemes, by having the attacker leverage interaction with the signing oracle to run the protocol.

But in our case there is a crucial type mismatch: we would like to build one-way functions, an inherently non-interactive primitive, by relying on secure MPC, which is an inherently interactive protocol. We resolve this mismatch by relying on obfuscation to *non-interactively implement* the next message function of each party in a secure MPC protocol[1]. Specifically, the output of our one-way function consists of an obfuscation of the next-message function for the appropriate MPC.

When sufficiently many one-way functions are combined, an adversary can execute an MPC protocol between them by appropriately querying their next message functions, as a result, inverting all one-way functions simultaneously with probability $\mu(\lambda)$.

But obfuscating the next message function in this manner exposes individual one-way functions (i.e., "participants" in the MPC protocol) to a new threat model. An adversary can query an obfuscated program repeatedly and in an arbitrary order, amounting to what are called "resetting" attacks [CGGM00]. This requires us to confront the need for MPC protocols that are secure against such strong resetting attacks.[2]

Resettable MPC. The question of whether secure MPC can be achieved in a setting where participants can be simultaneously reset has previously been studied by [GS09, GM11] and for the specific setting of zero knowledge in [CGGM00, BGGL01, BP13, COPV13, COP$^+$14, BP15, CPS16]. In particular, the work of [GS09] considered resetting attacks on only one party and obtained positive results. In the general setting where more than one party can be reset, [GM11] provided negative results for certain functionalities thereby ruling out a general purpose protocol for all functionalities. In addition, they obtained positive results for a limited class of entropic functionalities.

We observe that this negative result can be side-stepped by allowing our MPC simulator to run in super-polynomial time. Technically, we build on the recent concurrent secure MPC protocols

[1]This approach has previously been studied in multiple other contexts [GGHR14, DHRW16, AJN$^+$16]. However, in all those cases, the inputs to the MPC are fixed apriori and (implicitly) hardwired into the obfuscated programs. Looking ahead, in our protocol, the inputs cannot be fixed apriori and we resort to using resettable MPC to overcome this crucial issue. Note that this issue is also the reason why we do not know how to use iO to implement our obfuscated next-message function, whereas the earlier works mentioned above were able to use only iO.

[2]A similar issue also came up in [DJMW12] when the MPC was embedded in a signing oracle, which is interactive but stateless. It was resolved similarly by relying on some form of resettable MPC.

with super-polynomial simulation in [BGJ$^+$17], which are themselves based on the notion of super-polynomial strong simulation [KS17]. MPC security with super-polynomial simulation [Pas03, PS04] turns out to be sufficient for many scenarios, including for building our counterexamples.

As a contribution of independent interest, we obtain the first resettable MPC protocol for general functionalities, admitting super-polynomial simulation. Our protocol requires only three rounds of interaction, and assumes the existence of two-round sub-exponentially receiver-private and statistically sender-private OT, which can in turn be based on a variety of standard assumptions including the (sub-exponential) hardness of DDH, LWE, QRA, or DCRA [NP01, AIR01, HK12, BGI$^+$17, BD18, DGI$^+$19]. We state this result in the form of the following informal theorem.

Theorem 1.1. *(Informal) Assuming the existence of sub-exponentially receiver-private and statistically sender-private two-round OT, there exists a three-round resettably-secure MPC protocol for general functionalities, admitting a super-polynomial simulator.*

We stress that our resettable MPC protocol does not assume any form of obfuscation.

Our Counterexample to the Dream XOR Lemma. Armed with resettable MPC, we show that one-way functions with hard-core predicates to which the Dream XOR lemma provably does not apply can be obtained in the ideal obfuscation model, as follows: on input $x = (x_1||x_2) \in \{0,1\}^\lambda$, the one-way function f simply outputs an injective one-way function g applied to x_1, the first $\lambda/2$ bits of x, and uses the remaining $\lambda/2$ bits to obfuscate the next-message function of a participant in an MPC protocol. The ideal functionality for this MPC protocol obtains inputs (i.e., the x_1 values) from several participants, and with probability exactly $\mu(\lambda)$, outputs all these x_1 values in the clear. The hardcore bit of f on input $x = (x_1||x_2)$ is defined as the hardcore bit of g on input x_1.

We rely on security of the obfuscation scheme, the resettable MPC protocol and the one-way function g to argue that it is hard to recover x_1 (or predict the hard-core bit) of a single instance of this one-way function with probability significantly larger than $\mu(\lambda) \cdot \mathsf{poly}(\lambda) + \mathsf{negl}(\lambda)$. On the other hand, no matter how many times we repeat in parallel, the ability to execute a co-ordinated MPC program between all instances of the one-way function gives rise to an adversarial strategy that efficiently recovers all x_1 values, and therefore hardcore bits from all parallel instances, with probability at least $\mu(\lambda)$. For $\mu(\lambda) = 2^{-\sqrt{\lambda}}$, we prove the following informal theorem.

Theorem 1.2. *(Informal). Assuming resettably-secure MPC with super-polynomial simulation for general functionalities, for target negligible function $\mu(\lambda) = 2^{-\sqrt{\lambda}}$, there exists an explicit counterexample to the dream XOR lemma in the ideal obfuscation model. Furthermore, such a counterexample exists in the plain model under a plausible special-purpose obfuscation assumption.*

We choose to set $\mu(\lambda) = 2^{-\sqrt{\lambda}}$ primarily for the sake of simplicity in exposition, but our technique also generalizes to rule out arbitrary negligible $\mu(\lambda)$. While ideal obfuscation does not exist in the plain model [Had00, BGI$^+$01], the theorem applies relative to any world in which ideal obfuscation exists. This can refer to any *oracle* (in the complexity-theoretic sense) that enables ideal obfuscation, or given the ability to obfuscate functions using ideal trusted hardware. This counterexample is meaningful even when given ideal obfuscation, because all algorithms are given free access to the obfuscated function, and moreover the model does not introduce any shared randomness; as a result, instances of our one-way function remain *truly independent*.

Replacing Ideal Obfuscation with a Concrete Obfuscation Conjecture. In fact we go one step further and we postulate that a very specific functionality can be obfuscated, in a *virtual black-box* [BGI+01] (VBB) manner, *without auxiliary input.* As a result, assuming VBB or special-purpose obfuscation without auxiliary input for a specific class of circuits, we obtain counterexamples to the XOR lemma in the plain model.

How meaningful or plausible is this concrete assumption? While there are known impossibility results for VBB obfuscation of several functionalities, including PRFs *in the presence of auxiliary input*, the only known meaningful negative results on VBB without auxiliary input are the highly contrived "self-eating" programs developed by Barak et al. [BGI+01]. Despite it being plausible to embed the circuit family of [BGI+01] into *some* specific instantiations of resettable MPC and signatures, it appears extremely unlikely that *every* instantation of resettable MPC and signatures will have a [BGI+01]-style counterexample embedded into it. All we need for our counterexample is the existence of *one* VBB-obfuscatable family, which is compatible with all known evidence regarding VBB obfuscation. We stress again that our VBB assumption does not require security with respect to any auxiliary information. Indeed, such special-purpose obfuscation assumptions were used by Garg et al. [GGHW14] to prove negative results for differing-inputs obfuscation, and to this date, there are no known refutations of these types of conjectured assumptions for non-contrived circuits without auxiliary input.

Finally, we note that the recent work of [JLS21] has shown how to construct indistinguishability obfuscation (iO) from standard assumptions. In addition, several other constructions of iO from new assumptions that look plausible and are quite simple to state have appeared [Agr19, JLMS19, AJL+19, AP20, GJLS21, GP21, BDGM20a, BDGM20b, WW21, DQV+21]. While we do not know how to use indistinguishability obfuscation to achieve our result, this recent progress suggests that perhaps achieving VBB obfuscation for circuit families such as ours may also be possible from standard assumptions. Our work offers further motivation for this important line of study.

1.3 Second Counterexample

Our second counterexample begins with the work of [DJMW12], which constructs a counterexample to a dream version of *direct-product* hardness amplification. In particular, they construct a hard relation R such that, given a uniformly random instance $\widetilde{x}$, no polynomial time adversary can find a witness w such that $(\widetilde{x}, w) \in R$ except with negligible probability. However, given t independent copies $\widetilde{x}_1, \ldots, \widetilde{x}_t$, the adversarial advantage of finding all t witnesses w_i such that $(\widetilde{x}_i, w_i) \in R$ does not decrease much as t gets large. Concretely, there is a polynomial time adversary that can find all t witnesses with probability (say) $2^{-\sqrt{\lambda}}$, no matter how large t is. This counterexample is based on a non-standard hash function security property called extended second-preimage resistance (ESPR), which is weaker than collision resistance, but is assumed to hold for a fixed (un-keyed) hash function against non-uniform attackers. The work of [DJMW12] justifies this assumption by showing that ESPR security holds in the random-oracle model with auxiliary input [Unr07, DGK17], which models security properties for un-keyed hash functions with respect to non-uniform attackers.

As an initial idea to get a counterexample for the dream XOR lemma, one may hope to simply take a hard-core predicate for the relation R. The main issue is that the witness w for $\widetilde{x}$ is not unique, and so even if we are able to find *some* witness for $\widetilde{x}$, it does not mean we can compute *the* correct hard-core predicate. We resolve this by relying on an injective one way function $\hat{f}$ and a public-coins differing-inputs obfuscation (PCdiO) [ABG+13, BCP14, IPS15]. PCdiO is a strengthening of indistinguishability obfuscation (iO). All current constructions of iO can be conjectured to also satisfy PCdiO security, although no proofs of security for achieving PCdiO are as-yet known.

For the counterexample, we define a one-way function f that gets as input $x = (\hat{x}, \tilde{x}, r)$ and outputs $y = (\hat{y} = \hat{f}(\hat{x}), \tilde{x}, \tilde{C})$ where $\tilde{C}$ is an obfuscated circuit that takes as input a witness w, and if $(\tilde{x}, w) \in R$ it outputs $\hat{x}$, else $\perp$; we use r as the randomness for the obfuscation. We show that the function f is one-way and moreover, given the output y, it is hard to find $\hat{x}$. Intuitively, this follows because it is hard to find a valid witness w for $\tilde{x}$ that will make the obfuscated circuit output anything useful, and therefore the obfuscated circuit is indistinguishable from one that does not contain $\hat{x}$ and always outputs $\perp$; on the other hand the one-wayness of $\hat{f}$ says that it is hard to compute $\hat{x}$ from $\hat{y}$. We define a predicate $P(x)$ to be Goldreich-Levin hardcore bit of $\hat{x}$, and the above shows that $P(x)$ is a hard-core predicate of $f(x)$. On the other hand, it is easy to see that the dream XOR lemma does not hold for f, P. Given many values $y_i = f(x_i)$ for $i = 1, \dots, t$, we can find all the witnesses $w_1, \dots, w_t$ for $\tilde{x}_1, \dots, \tilde{x}_t$ with probability $2^{-\sqrt{\lambda}}$. We then input these witnesses w_i to the respective obfuscated programs contained in y_i to recover $\hat{x}_i$, which allows us to recover all the hardcore-predicates $P(x_i)$ and therefore also $\bigoplus P(x_i)$.

We obtain the following informal theorem.

Theorem 1.3. *(Informal). Assuming the existence of public-coins differing-inputs obfuscation (PCdiO), extended second-preimage resistant (ESPR) hash functions, and injective one-way functions there exists an explicit counterexample to the dream XOR lemma.*

We observe that we do not actually even need PCdiO for the above counterexample, and (public-coins) extractable witness encryption suffices; instead of obfuscating the circuit that takes as input a witness w, and if $(\tilde{x}, w) \in R$ it outputs $\hat{x}$, we use witness encryption to encrypt the message $\hat{x}$ with respect to the statement $\tilde{x}$ for the relation R.

1.4 Counterexamples for the Goldreich-Levin Predicate

We note that, in both our counterexamples, only the choice of the one-way function is "artificial", but the hardcore predicate is just the Goldreich-Levin (GL) predicate, albeit only applied to one of the components of the input, rather than the entire input as a whole. One may ask whether it is possible to get a counterexample where the hardcore predicate is GL applied to the entire input, or whether there is hope that the dream XOR lemma would hold in this case. Although we do not know how to get a counterexample for the GL predicate applied to the pre-image of a one-way function, we can get a counterexample if we generalize to *one-way puzzles* and consider the GL predicate applied to the solution of such a puzzle. In a one-way puzzle, there is a randomized algorithm that generates random hard puzzles together with a solution that can be verified in polynomial time. Security says that no polynomial time attacker can solve a random hard puzzle with better than negligible probability. In this case, we can take the one-way functions from our counterexamples and define a hard puzzle consisting of the one-way function output, while the solution is just the small component of the one-way function's input that we apply GL to. In both examples, we can efficiently verify the solution. To summarize, the above shows that the dream XOR lemma fails, even when restricted to the specific GL predicate, at least when applied to general one-way puzzles.

1.5 Related Work

We note that there is much work in cryptography on designing counter-examples to statements that "should intuitively hold" but don't. Even when it becomes clear that a proof for such statements is lacking, a counter-example provides some tangible understanding of how things can go wrong. Some such works that provide interesting counterexamples include: parallel repetition of multi-player games [For89, Fei91, HY19] and cryptographic protocols [BIN97, PW07], hardness amplification [DJMW12], circular security of encryption schemes [Rot13, KRW15, AP16, KW16, GKW17b, WZ17, GKW17a], selective opening attack security of encryption and commitments [BHY09, HR14, HRW16], leakage amplification via parallel repetition [LW10, JP11, DJMW12], hardness of prediction via obfuscation [BFM14]. Such counter-examples can point us to refined versions of the statement that may potentially still hold. Furthermore, exactly because such counter-examples "defy intuition", they often capture interesting ideas and techniques that turn out to be of greater value down the line. For example, the techniques developed in the context of circular security counter-examples [GKW17b] lead to positive results on obfuscation from LWE [WZ17, GKW17a].

2 Detailed Technical Overview of the First Counterexample

As discussed in the introduction, we obtain our counterexample by implementing a variant of Rudich's oracle in a completely decentralized manner, without introducing any correlations between instances of our counterexample. In this section, we outline our construction and give an overview of our proof technique.

A central cryptographic primitive that enables decentralized computation is secure multi-party computation (MPC). MPC enables several mutually distrusting participants to jointly compute a function f of their private inputs while only revealing the output y of f applied to their joint inputs, and revealing no information to each player beyond their own input and the output y.

2.1 XOR lemma counterexample

We design a one-way function $\mathcal{G}$ as our counterexample to the XOR lemma. $\mathcal{G}$ on input uniform randomness $x = (\alpha||\beta)$, outputs $f(\alpha)$ for an injective one-way function f, and uses randomness β to build a "proxy" that participates in an MPC protocol. This proxy is simply the next-message function of a participant in the appropriate MPC protocol. We define a hardcore predicate for $\mathcal{G}$ on input $x = (\alpha||\beta)$ as the output of a hardcore predicate for $f(\alpha)$.

The MPC protocol will allow multiple such proxies to jointly emulate a randomized ideal functionality that obtains the value α as input (from each participating proxy), and with probability $\mu(\lambda)$ for some fixed negligible function $\mu(\cdot)$, outputs all the α values it obtained as input from all proxies, and otherwise outputs $\perp$. If we are able to implement such an MPC protocol, it is clear that no matter how many times we repeat, an adversary would be able to run the MPC protocol between all instances of the one-way function and obtain the α values, and therefore the hardcore predicates of all instances, simultaneously, with probability at least $\mu(\lambda)$.

But in order for this to be a valid counterexample, we also need to ensure that the hard-core predicate for a *single* instance of this one-way function is secure: that is, it cannot be predicted with probability close to 1. In fact, we prove that the hard-core bit cannot be predicted with advantage better than $\mathsf{negl}(\lambda)$. For this, we must ensure that even given a next-message function that has the value α hardwired in it, it is not possible to extract α except with probability $\mathsf{negl}(\lambda)$. As a first step, we *obfuscate* the next-message function instead of releasing it in the clear. Assuming that the next-message circuit can be obfuscated with virtual black-box security, this restricts all information that can be learned from the obfuscated program to information that can be obtained via input-output access alone. Note, however, that we are not done yet, since this still allows an adversary, who has access to the obfuscated circuit, to query the next-message function multiple times on the same partial transcript, and potentially out of order. We use signatures to fix the latter issue and ensure that the adversary cannot make any meaningful queries on round $(i+1)$ of an MPC execution without querying on round i, for any $i \geq 1$. Unfortunately, this still leaves the obfuscated next-message circuit vulnerable to a "resetting" attack: where an adversary can query such a circuit to obtain multiple executions with the same partial transcript. Therefore, we need to harden our MPC protocol to obtain security against resetting attacks.

Finally, we point out one remaining (subtle) issue. Note that our counterexample is based on VBB-obfuscating the next-message function of a participant in an MPC protocol. In our setting, the adversary – given a single instance of our one-way function – can query this obfuscated program by generating his own next-message functions for imaginary MPC participants. Importantly, the adversary gets to *choose* the identity of these participants. But giving the adversary the freedom to choose the identity of a participant enables it to invoke the same participant (with the same input and randomness), many times using multiple identities in the same protocol. This could, for instance, allow the adversary to make multiple copies of the honest one-way function, and instantiate n parties, all having the same input and random tape, and potentially use this to manipulate the randomness of ideal functionality. More generally, when running an MPC protocol it is assumed that all players have different identities, and the adversary *does not* have the freedom to manipulate the identities of honest parties (or make multiple copies of any given honest identity).

In our setting, since every party is an obfuscated program, we must ensure that messages generated by each program are properly authenticated *under distinct verification keys*. We therefore add an explicit check to our VBB obfuscated program: the program will check that all verification keys are different and all messages in the input transcript are correctly signed. Ensuring that all verification keys are different guarantees that even in the ideal world, every participant of the MPC protocol has a different identity. Once distinct identities are carefully enforced in this way, the underlying MPC protocol (via underlying tools such as non-malleable commitments) ensures that the secret inputs of players to the MPC protocol are all independent of each other. Assuming the existence of resettable MPC, we describe our formal construction in Section 5.

Next, we turn to building MPC secure against resetting attacks. As discussed in the introduction, such resettably-secure MPC protocols are only known for ideal functionalities that satisfy a specific property [GM11][3]. Since our ideal functionality does not satisfy this property, we develop an appropriate resettable MPC protocol for use in our setting. We relax security to allow super-polynomial simulation, because that suffices for our applications. In what follows, we provide an overview of this protocol.

[3]We point out that [GM11] give resettably secure protocols with polynomial simulation, but only for (very limited) functionalities that satisfy a special property. Informally, they require that there exist compression and decompression algorithms such that with overwhelming probability, for all possible inputs, the compression algorithm generates a small suggestion string s which helps the decompression algorithm correctly predict the output of the adversary in any given session (given the adversary's input-output pairs in prior sessions and its current input, without knowledge of the honest party's input). We refer the reader to [GM11] for details and a formal description of this property.

2.2 Resettable MPC

Our construction of MPC with superpolynomial simulation (SPS), secure against resetting attacks, proceeds in two steps.

- **Resettably-Secure SPS MPC against Semi-malicious adversaries.** As a first step, we build resettably secure MPC against semi-malicious adversaries. A semi-malicious adversary always produces messages in the support of honestly generated messages, and writes the input and randomness used to generate these messages on a special witness tape. We compile a semi-malicious MPC protocol that is *not necessarily secure against resetting attacks* into one that is resettably secure against semi-malicious adversaries. Our compiler simply appends a round to the beginning of the protocol; in this round, each party P_i must commit to its input and to a uniformly sampled PRF key K_i. Next, all participants execute the semi-malicious resettably-insecure MPC protocol, except in every round, each party P_i uses randomness that is generated by evaluating the PRF with key K_i on the transcript so far.

  The effect of this is that all protocol messages sent by P_i become a deterministic function of the values committed in the first round and the transcript so far. In fact, *any* protocol transcript generated by any combination of honest and semi-malicious adversarial participants is a deterministic function of the first round.

  Therefore, any adversary that resets to a previous point in the protocol after round 1 and behaves semi-maliciously, must send exactly the same messages each time. On the other hand, any adversary that resets to the middle of round 1 sees an entirely new execution of the semi-malicious protocol. As a result, we are able to prove that the resulting resettably-secure semi-malicious MPC protocol admits a polynomial time simulator. Next, we compile to obtain an MPC protocol resettably-secure against fully malicious adversaries.

- **Resettably-Secure SPS MPC against Malicious Adversaries.** Our compiler is identical to the one in [BGJ$^+$17], which builds three round concurrent MPC with superpolynomial simulation. Here, in addition, we prove resettable security. The only difference is that while the compiler in [BGJ$^+$17] uses as subroutine an underlying stand-alone secure MPC against semi-malicious adversaries; we instead use a *resettably* secure MPC against semi-malicious adversaries (as described above). As ingredients, beyond the resettable semi-malicious protocol, this compiler relies on two round non-malleable commitments and two round zero knowledge arguments with super-polynomial strong simulation [KS17]. We note that all of these ingredients are secure (or can be easily modified to be secure) under resetting attacks. Furthermore, following [BGJ$^+$17], we show that the resulting protocol admits a straight-line superpolynomial time simulator. Resettable security of the resulting protocol against malicious adversaries follows by a careful analysis of this simulator and makes use of the resettable security of all ingredients. We formalize these ideas in section 4.

3 Preliminaries

Throughout the paper, let λ denote the security parameter.

3.1 Virtual Black Box Obfuscation

We recall the definition of Virtual Black-Box (VBB) obfuscation from Barak et al.[BGI$^+$01]. In this definition, we don't allow any auxiliary inputs (therefore making our assumptions weaker).

Definition 3.1 (Virtual Black-Box Obfuscation)**.** *For any polynomial $t(\cdot)$, circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, a uniform PPT oracle machine* Obf *is a "Virtual Black-Box" Obfuscator for $\mathcal{C}$ if the following conditions are satisfied:*

- *Functionality: For every $\lambda \in \mathbb{N}$ and every input x:*

$$\Pr[(\mathsf{Obf}(C))(x) \neq C(x)] \leq \mathsf{negl}(|C|),$$

  where the probability is over the coins of $C \leftarrow \mathcal{C}_\lambda$.

- *Polynomial Slowdown: There exists a polynomial $p(\cdot)$ such that for every $\lambda \in \mathbb{N}$ and every $C \in \mathcal{C}_\lambda$, $|\mathsf{Obf}(C)| \leq p(|C|)$.*

- *Virtual Black-Box: For every PPT adversary $\mathcal{A}$ there exists a PPT simulator* $\mathsf{Obf.Sim}$*, and a negligible function μ such that for every $\ell \in \mathbb{N}$ and every $C \in \mathcal{C}_\ell$:*

$$\left|\Pr\left[\mathcal{A}\left(\mathsf{Obf}(C)\right) = 1\right] - \Pr\left[\mathsf{Obf.Sim}^{C}\left(1^{|C|}\right) = 1\right]\right| \leq \mu(|C|) \ ,$$

  where the probabilities are over the coins of $\mathsf{Obf.Sim}$ *and* Obf*.*

3.2 Statistically Sender Private OT

We rely on a 1-out-of-2 *Oblivious Transfer* protocol where one party, the *sender*, has input composed of two strings (M_0, M_1) and the input of the second party, the *receiver*, is a bit c. The receiver should learn M_c and no information about M_{1-c} while the sender should gain no information about c. We give a definition for the setting where the sender is protected information theoretically while the receiver is protected only computationally.

Definition 3.2 (Statistically Sender Private OT.)**.** *The receiver runs the algorithm OT_1 which takes 1^λ and a choice bit $c \in \{0,1\}$ as input and outputs $(\mathsf{ot}_1, state)$. The receiver then sends ot_1 to the sender, who obtains ot_2 by evaluating $\mathsf{OT}_2(1^\lambda, \mathsf{ot}_1, M_0, M_1)$, where M_0 and M_1 (such that $M_0, M_1 \in \{0,1\}^\lambda$) are its inputs. The sender then sends ot_2 to the chooser who obtains M_c by evaluating $\mathsf{OT}_3(1^\lambda, \mathsf{ot}_2, state)$.*

- ***Perfect correctness.*** *For every choice bit $c \in \{0,1\}$ of the chooser and input messages M_0 and M_1 in $\{0,1\}^\lambda$ of the sender we require that, if $(\mathsf{ot}_1, state) \leftarrow \mathsf{OT}_1(1^\lambda, c)$, $\mathsf{ot}_2 \leftarrow \mathsf{OT}_2(1^\lambda, \mathsf{ot}_1, M_0, M_1)$, then $\mathsf{OT}_3(1^\lambda, \mathsf{ot}_2, state) = M_c$ with probability 1.*

- ***Receiver's security.*** *We require that for every polynomial-size adversary $\mathcal{A}$, $|\Pr[\mathcal{A}(\mathsf{OT}_1(1^\lambda, 0)) = 1] - \Pr[\mathcal{A}(\mathsf{OT}_1(1^\lambda, 1)) = 1]|$ is negligible in λ.*

- ***Statistical sender's security.*** *We define an unbounded[4] time extractor OTExt such that OTExt on any input ot_1 outputs 0 if there exists some random coins such that $\mathsf{OT}_1(1^\lambda, 0)$ outputs ot_1, and 1 otherwise.*

[4]Note that by fixing the parameters of the scheme, we can bound the running time of the extractor by some sub-exponential function but we avoid that to keep notation simple and avoid unnecessary parameters.

Then for any value of ot_1, *and any* K_0, K_1, L_0, L_1 *with* $K_{\mathsf{OTExt}(\mathsf{ot}_1)} = L_{\mathsf{OTExt}(\mathsf{ot}_1)}$, *we have that* $\mathsf{OT}_2(1^\lambda, \mathsf{ot}_1, K_0, K_1)$ *and* $\mathsf{OT}_2(1^\lambda, \mathsf{ot}_1, L_0, L_1)$ *are statistically indistinguishable.*

We will, in fact require the receiver's security property described above to hold against all subexponential-sized circuits. Specifically, we will require that there exist a constant $c > 0$ such that for every circuit $\mathcal{A}$ of size 2^{λ^c}, $|\Pr[\mathcal{A}(\mathsf{OT}_1(1^\lambda, 0)) = 1] - \Pr[\mathcal{A}(\mathsf{OT}_1(1^\lambda, 1)) = 1]|$ is negligible in λ.

3.3 Resettable MPC

In Appendix A, we define the notion of Multiparty Computation (MPC) that is resettably secure similar to the definition in Goyal and Maji [GM11]. The difference in that, in our setting, we consider a simulator that can run in super-polynomial time [Pas03, BS05].

Remark. We note that security holds only when the identities of all the parties in the MPC protocol are distinct.

4 Resettable MPC

In this section, we construct a three round resettably secure MPC protocol based on sub-exponentially secure DDH or LWE. Formally, we show the following theorem:

Theorem 4.1. *Assuming sub-exponentially secure* 2 *round statistically sender private and sub-exponentially receiver private OT according to Definition 3.2, there exists a three round resettable secure MPC protocol with super-polynomial simulation, for any functionality satisfying Definition A.1.*

Our construction proceeds in two stages. First, assuming 2 round OT, we construct an MPC protocol π^{RSM} that is resettably secure against adversary that is malicious in the first round and semi-malicious in subsequent rounds. Next, assuming 2 round OT, we show how to compile a sub-exponentially secure version of this protocol π^{RSM} into a protocol that is resettably secure against malicious adversaries (under super-polynomial simulation).

4.1 Construction: Semi-Malicious Security

In this section, we describe a compiler that transforms any k-round semi-malicious MPC protocol that may or may not be resettable secure, into one that is semi-malicious secure under resetting attacks. Formally, we show the following:

Theorem 4.2. *For any* $k > 0$, *assuming the existence of a* k*-round MPC protocol secure against semi-malicious adversaries against all but one corruptions, there exists a* $(k+1)$*-round MPC protocol that is resettably secure against adversaries that behave maliciously in the first round and semi-malicious in the remaining rounds, and corrupt upto all but one parties.*

Instantiating this protocol with the 2 round MPC protocol of Garg and Srinivasan [GS18] or Benhamouda and Lin [BL18], which can be instantiated assuming 2 round OT (which in turn can be obtained from DDH or LWE), we have the following corollary:

Corollary 4.3. *Assuming the hardness of DDH or LWE, there exists a three round MPC protocol for any functionality* f *that is resettably secure against adversaries that can behave maliciously in the first round and semi-maliciously in the remaining rounds, and corrupt up to all but one parties.*

Notation and Primitives Used: Let π be an n party MPC protocol over a broadcast channel that is secure against (non-resetting) semi-malicious adversaries. Let $\pi.\mathsf{NextMsg_j}$ denote the algorithm used by any party to compute its message in round j and let $p(\lambda)$ denote the length of the randomness used by the algorithm. Let $\pi.\mathsf{OUT}$ denote the algorithm to compute the final output. Let the number of rounds be ℓ. Let $\tau(\lambda)$ denote the maximum length of the transcript of an execution of π when each party uses inputs of length λ. Let $\mathcal{S} = (\mathcal{S}_1, \ldots, \mathcal{S}_\ell)$ denote the straight line simulator for the above protocol π - that is, $\mathcal{S}_i$ is the simulator's algorithm to compute the i^{th} round messages. Let $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^{\tau(\lambda)} \rightarrow \{0,1\}^{p(\lambda)}$ be a pseudorandom function.

Protocol:
Let f be any functionality. Consider n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ with inputs $\mathsf{x}_1, \ldots, \mathsf{x}_n$ respectively who wish to evaluate f on their joint inputs. Let $(\mathsf{k}_i, \mathsf{r}_i^{\mathsf{Com}})$ denote the randomness of each party P_i where $|\mathsf{k}_i| = \lambda$ and $|\mathsf{r}_i^{\mathsf{Com}}| = \lambda^2$. The protocol π^{RSM} proceeds as follows:

1. **Round 1:**
   Each party P_i does the following:
   - Sample a key $\mathsf{k}_i \leftarrow \{0,1\}^\lambda$ for the function PRF.
   - Compute and send $\mathsf{msg}_{1,i} \leftarrow \mathsf{Commit}(\mathsf{k}_i || \mathsf{x}_i)$.
2. **Rounds $2 \ldots \ell$:**
   Let the round number be j. Let Trans_{j-1} denote the protocol transcript after round $(j-1)$. Each party P_i does the following:
   - Compute and send $\mathsf{msg}_{j,i} \leftarrow \pi.\mathsf{NextMsg_j}(\mathsf{x}_i, \mathsf{Trans}_{j-1}; \mathsf{r}_{j,i})$ where the randomness $\mathsf{r}_{j,i} = \mathsf{PRF}(\mathsf{k}_i, \mathsf{Trans}_{j-1})$.
3. **Output Computation:**
   Each party P_i does the following:
   - Let Trans_ℓ denote the transcript of the underlying protocol π at the end of its execution.
   - Compute and output $\mathsf{y}_i \leftarrow \pi.\mathsf{OUT}(\mathsf{x}_i, \mathsf{Trans}_\ell)$.

We defer the security proof to Appendix B.1.

4.2 Construction: Malicious Security

We now show how to compile the semi-malicious secure MPC protocol π^{RSM} from the previous section into a protocol that is resettably secure against malicious adversaries with super-polynomial simulation. Formally, we show the following:

Theorem 4.4. *Assuming :*

- *Sub-exponentially secure DDH/LWE, and,*
- *a 3 round MPC protocol for any functionality f that is resettably secure against adversaries that can behave maliciously in the first round and semi-maliciously in the remaining two rounds and corrupt up to all but one parties,*

there exists a 3 *round MPC protocol for any functionality f that is resettably secure under super-polynomial simulation against malicious adversaries that can corrupt up to all but one parties, according to Definition A.1.*

Instantiating the underlying three round MPC protocol in the above theorem with the protocol π^{RSM} from the previous section, we achieve Theorem 4.1.

Our construction is identical to the one in [BGJ$^+$17], which builds three round concurrent MPC with superpolynomial simulation. Here, we prove resettable security of the same construction. As ingredients, beyond the underlying resettable semi-malicious protocol, this compiler relies on two round non-malleable commitments and two round zero knowledge arguments with super-polynomial strong simulation [KS17]. We note that both these ingredients are secure under resetting attacks. We provide the complete protocol and proof in Appendix B.2.

5 Counterexample via VBB Obfuscation and Resettable MPC

We start this section by defining hard-core predicates. Next, we state the dream version of Yao's XOR lemma, and then we describe our counterexample.

Definition 5.1. *($\epsilon(\lambda)$-Hard Core Predicate.) Let λ denote a security parameter and $m = m(\lambda), n = n(\lambda)$ be polynomials in λ. An $\epsilon(\lambda)$-hard core predicate of a one-way function $f : \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ is a predicate $P : \{0,1\}^{m(\lambda)} \to \{0,1\}$ such that for every polynomial-time non-uniform $\mathcal{A}$ and every λ,*

$$\Pr_{x \leftarrow \{0,1\}^\lambda}[\mathcal{A}(1^\lambda, f(x)) = P(x)] \leq \frac{1}{2} + \epsilon(\lambda)$$

Lemma 5.2 (Hard Core Predicate for a One-Way Function)**.** *[GL89] If f is a one-way function, then $h(x,r) = \langle x, r\rangle(\mathsf{modulo}\ 2)$ is an $\epsilon(\lambda)$-hard core predicate, according to Definition 5.1, for the one-way function f' defined by $f'(x,r) = (f(x), r)$, for some $\epsilon(\lambda) = \mathsf{negl}(\lambda)$.*

Conjecture 5.3 (Dream version of Yao's XOR Lemma)**.** *[BGI08] Fix $\mu(\lambda) = 2^{-\sqrt{\lambda}}$. Let f denote a one-way function and P denote an $\epsilon(\lambda)$-hard core predicate according to Definition 5.1. Then for any $t = \mathsf{poly}(\lambda)$, $P^{(t)}(x_1, \ldots, x_t) = \bigoplus_{i\in[t]} P(x_i)$ is an $\epsilon'(\lambda)$-hard core predicate for $f'(x_1||x_2||\ldots||x_t) \triangleq f(x_1)||f(x_2)||\ldots||f(x_t)$, such that $\epsilon'(\lambda) \leq \epsilon(\lambda)^{t(\lambda)} + \mu(\lambda)$.*

We note that this conjecture is a special case of (and is therefore implied by) the dream conjecture first formulated in [GNW11]. Also, we fixed $\mu(\lambda)$ to be an arbitrary negligible function in λ, specifically, we set it to $2^{-\sqrt{\lambda}}$ for ease of exposition. However, we note that our refutation of this conjecture can be generalized to refute arbitrary negligible functions $\mu(\cdot)$ by setting the parameters of our one-way function accordingly. That is, our proof can be generalized to show that for every negligible function $\nu(\cdot)$, there exists a one-way function that refutes Conjecture 5.3 with $\mu(\cdot) = \nu(\cdot)$. We will now construct a one-way function for which Conjecture 5.3 does not hold.

5.1 Construction

We define a one-way function $\mathcal{G} : \{0,1\}^{2\lambda} \to \{0,1\}^{p'(\lambda)}$, where $p'(\cdot)$ is a polynomial in λ, the exact value of which will be determined later.

Notation and Primitives Used.

- Let $g : \{0,1\}^\lambda \rightarrow \{0,1\}^{p(\lambda)}$ be any injective one way function and h denote the Golreich-Levin [GL89] hardcore bit for this one way function.
- Let π^{Res} denote any resettably secure MPC protocol with superpolynomial simulation. Let $(\pi^{\mathsf{Res}}.\mathsf{NextMsg}_1, \pi^{\mathsf{Res}}.\mathsf{NextMsg}_2, \ldots, \pi^{\mathsf{Res}}.\mathsf{NextMsg}_\mathsf{n})$ denote the algorithms used by each party to compute the messages in each of the rounds and $\pi^{\mathsf{Res}}.\mathsf{OUT}$ denote the algorithm used by each party to compute its final output. Also, let Trans_i denote all messages sent in an execution of π^{Res} up to round i. Let $\mathsf{Res.Sim}$ denote the straight-line (super-polynomial) simulator for this protocol. Let $(\mathsf{Res.Sim.NextMsg}_1, \ldots, \mathsf{Res.Sim.NextMsg}_\mathsf{n})$ denote the algorithms used by the simulator to compute the messages in each of the rounds and $\mathsf{Res.Sim.Out}$ denote the algorithm used to compute the final output on behalf of the honest parties. Let $\ell(\lambda)$ denote the length of the randomness required by each party on inputs of length λ. Let $s(\lambda)$ denote the maximum size of the circuit representation of $(\pi^{\mathsf{Res}}.\mathsf{NextMsg}_1, \pi^{\mathsf{Res}}.\mathsf{NextMsg}_2, \ldots, \pi^{\mathsf{Res}}.\mathsf{NextMsg}_\mathsf{n}, \pi^{\mathsf{Res}}.\mathsf{OUT})$ in this protocol.
- Let $(\mathsf{Obf}, \mathsf{Eval})$ be a VBB obfuscation scheme and $\mathsf{Obf.Sim}$ denote its simulator. Let $r(\lambda)$ denote the length of the randomness used to obfuscate programs of size $s(\lambda)$, and $s'(\lambda)$ denote the size of the obfuscated program.
- Let $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a sub-exponentially unforgeable signature scheme.
- Let $\mathsf{PRG} : \{0,1\}^\lambda \rightarrow \{0,1\}^{\sqrt{\lambda}+\lambda+r(\lambda)+\ell(\lambda)}$ be a pseudorandom generator.

Construction. The one-way function $\mathcal{G}$, on input $x = (a, b)$, where $|a| = |b| = \lambda$, is:

- Compute $y = g(a)$.
- Compute $(\alpha, \beta, \gamma, \delta) \leftarrow \mathsf{PRG}(b)$ where α is of length $\sqrt{\lambda}$, β is of length λ, γ is of length $\ell(\lambda)$ and δ is of length $r(\lambda)$. Set $\mathsf{r}^{\mathsf{Res}} = \gamma$.
- Generate $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda; \beta)$.
- Compute $\widehat{\mathsf{C}} = \mathsf{Obf}(1^\lambda, \mathsf{C})$ using randomness δ where program C is described in Figure 1.
- Output $(\widehat{\mathsf{C}})$. We set $p'(\lambda) = |\widehat{\mathsf{C}}|$.

Hardcore Predicate Recall that the Goldreich-Levin hardcore predicate [GL89] for the one-way function g is h. We now define the hardcore predicate H for $\mathcal{G}$ as: $\mathsf{H}(x) = h(a)$, where $x = (a, b)$ such that $|a| = |b|$.

5.2 Security

We conjecture the following about the existence of special-purpose obfuscation.

Conjecture 5.4 (Special-purpose obfuscation)**.** *There exists a secure signature scheme and a resettable MPC protocol according to Definition A.1 for which, for the class of circuits* $\mathsf{C} = \{\mathsf{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *where for* $\lambda \in \mathbb{N}$, C_λ *is depicted in Figure 1, where* $a \in \{0,1\}^\lambda, y \in \{0,1\}^{p(\lambda)}, \alpha \in \{0,1\}^{\sqrt{\lambda}}, (\mathsf{SK}, \mathsf{VK}) \in \mathsf{Supp}(\mathsf{Gen}(1^\lambda)), \mathsf{r}^{\mathsf{Res}} \in \{0,1\}^{\ell(\lambda)}$, *there exists a virtual black-box obfuscator according to Definition 3.1.*

```
Hardwired values: $(a, y, \alpha, \mathsf{sk}, \mathsf{vk}, \mathsf{r}^{\mathsf{Res}})$
  1. Output $y$. Additionally, do the following.
  2. If input = (“$\mathsf{Reveal\ Identity\ for\ MPC}$”): output $\mathsf{vk}$.
  3. If input = (“$\mathsf{MPC}$”, “$\mathsf{Begin}$”, “$\mathsf{Party}$” $\mathsf{j}, \overrightarrow{\mathsf{vk}}$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
       • Output $\sigma = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||j))$
  4. If input = (“$\mathsf{MPC}$”, “$\mathsf{Round}$” $1$, “$\mathsf{Party}$” $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \sigma$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
           – $\mathsf{Verify}(\mathsf{vk}, \sigma, (\overrightarrow{\mathsf{vk}}||j)) \neq 1$.
       • Begin resettable MPC protocol $\pi^{\mathsf{Res}}$ amongst $\lambda^2$ parties $\mathsf{P}_1, \ldots, \mathsf{P}_{\lambda^2}$ playing the role of
         $\mathsf{P}_j$ with identity $\mathsf{vk}$. Use input $\alpha$, randomness $\mathsf{r}^{\mathsf{Res}}$ to evaluate functionality $f$ (Figure 2).
         Let $\ell$ be the number of rounds of $\pi^{\mathsf{Res}}$.
       • Compute $\mathsf{msg}_1 = \pi^{\mathsf{Res}}.\mathsf{NextMsg}_1(1^\lambda, \alpha; \mathsf{r}^{\mathsf{Res}})$, $\sigma_1 = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$.
       • Output $(\mathsf{msg}_1, \sigma_1)$.
  5. If input = (“$\mathsf{MPC}$”, “$\mathsf{Round}$” $\mathsf{i}$, “$\mathsf{Party}$” $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \overrightarrow{\mathsf{msg}}_{i-1}, \sigma_{i-1}, \mathsf{Trans}_{i-2}$):
       • Let $\mathsf{Trans}_{i-2}$ denote the transcript till round $(i-2)$ and $\overrightarrow{\mathsf{msg}}_{i-1}$ denote the set of
         messages sent by all parties in round $(i-1)$ of protocol $\pi$.
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
           – $\mathsf{Verify}(\mathsf{vk}, \sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j)) \neq 1$.
       • If round number $i \leq \ell$, do:
           – Continue $\pi^{\mathsf{Res}}$. Compute $\mathsf{msg}_i = \pi^{\mathsf{Res}}.\mathsf{NextMsg_i}(\mathsf{Trans}_{i-2}, \overrightarrow{\mathsf{msg}}_{i-1}, \alpha; \mathsf{r}^{\mathsf{Res}})$.
           – Set $\mathsf{Trans}_{i-1} = (\mathsf{Trans}_{i-2}||\overrightarrow{\mathsf{msg}}_{i-1})$, $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}||\mathsf{msg}_i||\ i||j))$.
           – Output $(\mathsf{msg}_i, \sigma_i)$.
       • If round number $i = (\ell + 1)$, do:
           – Compute $z = \pi^{\mathsf{Res}}.\mathsf{OUT}(\mathsf{Trans}_{\ell-1}, \overrightarrow{\mathsf{msg}}_\ell, \alpha; \mathsf{r}^{\mathsf{Res}})$.
           – If $z = 0$, output $a$. Else, output $\perp$.
```

Figure 1: Description of program C

Theorem 5.5. *Assuming Conjecture 5.4 and resettably secure MPC with super-polynomial simulation against malicious adversaries according to Definition A.1, Conjecture 5.3 is false. In particular, assuming Conjecture 5.4, and either sub-exponential DDH or LWE, Conjecture 5.3 is false.*

Input: For each $i \in [\lambda^2]$, party P_i has input α_i of length $\sqrt{\lambda}$. **Output:** If $(\alpha_1 \oplus \ldots \oplus \alpha_{\lambda^2}) = 0^{\sqrt{\lambda}}$, output 0. Otherwise, output $\perp$.

Figure 2: Description of Functionality f

This also implies that the above result holds in the ideal obfuscation model. Here we model obfuscation as an ideal functionality with two interfaces. An "Obfuscate" interface takes as input a program P and outputs a handle $\tilde{P}$. The handle can simply be a counter which is incremented on each invocation. The ideal functionality keeps a list of such tuples $(\tilde{P}, P)$. An "Evaluate" interface takes as input a handle $\tilde{P}$ and an input x, and finds the corresponding tuple $(\tilde{P}, P)$ in the list; if such a tuple exists it outputs $P(x)$ else $\perp$. We rely on the fact that Conjecture 5.4 holds relative to this oracle and that our counter-example uses the obfuscator in a black-box manner. This gives us the following corollary.

Corollary 5.6. *Assuming sub-exponential DDH or LWE, Conjecture 5.3 is false in the ideal obfuscation model.*

The rest of this section is devoted to the proof of this theorem. First, we prove that H defined above is indeed a hardcore predicate for function $\mathcal{G}$. We observe that for our construction of $\mathcal{G}$ and H, this also automatically proves that $\mathcal{G}$ is a one way function. This is for the following reason: suppose for contradiction that $\mathcal{G}$ is not a one way function. Then, there exists a PPT adversary that, given a value $\mathcal{G}(x = (a, b)) = (\widehat{\mathsf{C}})$, can compute an inverse $x' = (a', b')$ with non-negligible probability. However, observe that since g is an injective one way function, $a' = a$. Further, since $\mathsf{H}(x = (a, b)) = h(a)$, $\mathcal{A}$ can compute $\mathsf{H}(x') = \mathsf{H}(x)$. Thus, given just $\mathcal{G}(x)$, $\mathcal{A}$ can compute $\mathsf{H}(x)$ with non-negligible probability which contradicts the fact that H is a hardcore predicate for function $\mathcal{G}$. Therefore, it suffices to formally show that for every PPT adversary $\mathcal{A}$,

$$\mathsf{Pr}[\mathcal{A}(\mathcal{G}(x)) = \mathsf{H}(x)] = \frac{1}{2} + \mathsf{negl}(\lambda)$$

where the probability is over the randomness of x and $\mathcal{A}$. We do this via the following series of computationally indistinguishable hybrids $\mathsf{Hyb}_0, \ldots, \mathsf{Hyb}_6$.

Hyb_0 – **Real Experiment:** Sample $x = (a, b) \leftarrow \{0, 1\}^{2k}$. Compute $\mathcal{G}$ exactly as in the function description above to obtain $(\widehat{\mathsf{C}}) = \mathcal{G}(x)$, and output $\mathcal{A}(\widehat{\mathsf{C}}) \oplus h(a)$.

Hyb_1 – **Uniform Randomness:** Sample $(\alpha, \beta, \gamma, \delta) \leftarrow \{0, 1\}^{\sqrt{\lambda}+\lambda+\ell(\lambda)+r(\lambda)}$ uniformly as opposed to $(\alpha, \beta, \gamma, \delta) = \mathsf{PRG}(b)$ as in Hyb_0. Then compute $(\widehat{\mathsf{C}})$ the same way as Hyb_0. Output $\mathcal{A}(\widehat{\mathsf{C}}) \oplus h(a)$.

Hyb_2–**Simulate Obfuscation:** This is the same as Hyb_1, in particular $(\alpha, \beta, \gamma, \delta) \leftarrow \{0, 1\}^{\sqrt{\lambda}+\lambda+\ell(\lambda)+r(\lambda)}$. However, unlike Hyb_1, output $\mathsf{Obf.Sim}^{\mathsf{C}}(1^{|\mathsf{C}|}) \oplus h(a)$.

Hyb_3 – **Reject transcripts that were not previously signed:** The output of this hybrid is $\mathsf{Obf.Sim}^{\mathsf{C}_1}(1^{|\mathsf{C}_1|}) \oplus h(a)$ where the *stateful* functionality C_1 is described in Figure 3. We note that C_1 is the same as C, except that it stores the list of signatures it sends out and outputs "$\mathsf{Special\ Abort}$" if it obtains as input a valid signature outside this list. The differences from C are highlighted in red.

```
Hardwired values: $(a, y, \alpha, \mathsf{sk}, \mathsf{vk}, \mathsf{r}^{\mathsf{Res}})$
Initialize list $\mathcal{L} = \emptyset$.
  1. Output $y$. Additionally, do the following.
  2. If input = (“Reveal Identity for MPC”): output vk.
  3. If input = (“MPC”, “Begin”, “Party” j, $\overrightarrow{\mathsf{vk}}$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k\in[\lambda^2]}$ are not distinct,
       • Output $\sigma = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||j))$. Add $(\sigma, (\overrightarrow{\mathsf{vk}}||j))$ to $\mathcal{L}$.
  4. If input = (“MPC”, “Round” 1, “Party” j, $\overrightarrow{\mathsf{vk}}, \sigma$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k\in[\lambda^2]}$ are not distinct,
           – $\mathsf{Verify}(\mathsf{vk}, \sigma, (\overrightarrow{\mathsf{vk}}||j)) \neq 1$.
       • Output “Special Abort” if $(\sigma, (\overrightarrow{\mathsf{vk}}||j)) \notin \mathcal{L}$.
       • Begin protocol $\pi^{\mathsf{Res}}$ amongst $\lambda^2$ parties $\mathsf{P}_1, \ldots, \mathsf{P}_{\lambda^2}$ playing the role of $\mathsf{P}_j$ with identity
         vk. Use input $\alpha$, randomness $\mathsf{r}^{\mathsf{Res}}$ to evaluate functionality $f$ defined in Figure 2.
       • Compute $\mathsf{msg}_1 = \pi^{\mathsf{Res}}.\mathsf{NextMsg}_1(1^\lambda, \alpha; \mathsf{r}^{\mathsf{Res}})$, $\sigma_1 = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$. Add
         $(\sigma_1, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$ to $\mathcal{L}$.
       • Output $(\mathsf{msg}_1, \sigma_1)$.
  5. If input = (“MPC”, “Round” i, “Party” j, $\overrightarrow{\mathsf{vk}}, \overrightarrow{\mathsf{msg}}_{i-1}, \sigma_{i-1}, \mathsf{Trans}_{i-2}$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k\in[\lambda^2]}$ are not distinct,
           – $\mathsf{Verify}(\mathsf{vk}, \sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j)) \neq 1$.
       • Output “Special Abort” if $(\sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j))) \notin \mathcal{L}$.
       • If round number $i \leq \ell$, do:
           – Continue protocol $\pi^{\mathsf{Res}}$. Compute $\mathsf{msg}_i = \pi^{\mathsf{Res}}.\mathsf{NextMsg}_\mathsf{i}\,(\mathsf{Trans}_{i-2}, \overrightarrow{\mathsf{msg}}_{i-1}, \alpha; \mathsf{r}^{\mathsf{Res}})$.
             Set $\mathsf{Trans}_{i-1} = (\mathsf{Trans}_{i-2}||\overrightarrow{\mathsf{msg}}_{i-1})$.
           – Compute $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}||\mathsf{msg}_i||\ i||j))$.
           – Add $(\sigma_i, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}\ ||\mathsf{msg}_i||\ i||j))$ to $\mathcal{L}$.
           – Output $(\mathsf{msg}_i, \sigma_i)$.
       • If round number $i = (\ell + 1)$, do:
           – Compute $z = \pi^{\mathsf{Res}}.\mathsf{OUT}(\mathsf{Trans}_{\ell-1}, \overrightarrow{\mathsf{msg}}_\ell, \alpha; \mathsf{r}^{\mathsf{Res}})$.
           – If $z = 0$, output $a$. Else, output $\perp$.
```

Figure 3: Description of program C_1

Hyb_4 – **Simulate MPC:** The output of this hybrid is $\mathsf{Obf.Sim}^{\mathsf{C}_2}(1^{|\mathsf{C}_2|}) \oplus h(a)$ where the *stateful* functionality C_2 is described in Figure 4. We note that C_2 is the same as C_1, except that it executes the (superpolynomial) simulator $\mathsf{Res.Sim}$ for the resettable MPC protocol π^{Res}, where $\mathsf{Res.Sim}$ interacts with the ideal functionality from Figure 2. We note that this hybrid runs in superpolynomial time because the running time of C_2 is superpolynomial.

Hyb_5 – **Remove Input:** The output of this hybrid is $\mathsf{Obf.Sim}^{\mathsf{C}_3}(1^{|\mathsf{C}_3|}) \oplus h(a)$, where C_3 (Figure 5) is the same as C_2, except that the program C_3 no longer uses the pre-image a (it only has the output $y = g(a)$ hardwired, but not a). That is, Step 4 outputs $\perp$ instead of outputting a if if $z = 0$.

Hyb_6 – **Honest MPC:** The output of this hybrid is $\mathsf{Obf.Sim}^{\mathsf{C}_4}(1^{|\mathsf{C}_4|}) \oplus h(a)$ for C_4 from Figure 6. We note that C_4 is the same as C_3, except that it runs the resettable MPC protocol π^{Res} inside the program C_3 using the honest parties' code (whereas C_3 runs the simulator $\mathsf{Res.Sim}(\cdot)$ for resettable MPC). We also note that this hybrid runs in polynomial time unlike the previous two hybrids.

We first show that that the advantage of the adversary in every pair of consecutive hybrids is negligible.

Lemma 5.7. *Assuming the security of the pseudorandom generator* PRG*,*

$$\Big|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]\Big| = \mathsf{negl}(\lambda)$$

Proof. We show that if there exists a polynomial $p(\cdot)$ such that $\Big|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]\Big| \geq \frac{1}{p(\lambda)}$, then there exists a PPT adversary $\mathcal{A}_{\mathsf{PRG}}$ that with oracle access to $\mathcal{A}$, contradicts security of the pseudorandom generator.

$\mathcal{A}_{\mathsf{PRG}}$ gets a string η from an external challenger, which is either the output of the pseudorandom generator PRG on a random seed, or is sampled uniformly at random. $\mathcal{A}_{\mathsf{PRG}}$ samples $a \leftarrow \{0,1\}^k$, sets $(\alpha, \beta, \gamma, \delta) = \eta$ and generates the remaining messages for its interaction with $\mathcal{A}$ exactly as in Hyb_0. Now, if η were generated as the output of PRG, the interaction between $\mathcal{A}_{\mathsf{PRG}}$ and $\mathcal{A}$ corresponds exactly to Hyb_0 and if η were uniformly random, the interaction between $\mathcal{A}_{\mathsf{PRG}}$ and $\mathcal{A}$ corresponds exactly to Hyb_1. $\mathcal{A}_{\mathsf{PRG}}$ outputs $s \oplus h(a)$, where s is the output of $\mathcal{A}$ in the experiment. This leads to a contradiction, completing the proof of the lemma. □

Lemma 5.8. *Assuming* $(\mathsf{Obf}, \mathsf{Eval})$ *satisfies Conjecture 5.4,*

$$\Big|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]\Big| = \mathsf{negl}(\lambda)$$

Proof. Suppose the lemma is false. Then there exists non-negligible $\mu(\cdot)$ such that:

$$\Big|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]\Big| \geq \mu(\lambda)$$

which implies that over the randomness of sampling $(a, \alpha, \beta, \gamma, \delta)$ uniformly and sampling the circuit $C_a := \mathsf{C}_1$ according to Figure 1,

$$\Big|\Pr[\mathcal{A}(\mathsf{Obf}(C_a)) \oplus h(a) = 1] - \Pr[\mathsf{Obf.Sim}(1^{|C_a|}) \oplus h(a) = 1]\Big| \geq \mu(\lambda)$$

```
Hardwired values: $(a, y, \alpha, \mathsf{sk}, \mathsf{vk}, \mathsf{r}^{\mathsf{Res}})$
Initialize list $\mathcal{L} = \emptyset$.
  1. Output $y$. Additionally, do the following.
  2. If input = (“Reveal Identity for MPC”): output vk.
  3. If input = (“MPC”, “Begin”, “Party” $\mathsf{j}, \overrightarrow{\mathsf{vk}}$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
       • Output $\sigma = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||j))$. Add $(\sigma, (\overrightarrow{\mathsf{vk}}||j))$ to $\mathcal{L}$.
  4. If input = (“MPC”, “Round” 1, “Party” $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \sigma$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
           – $\mathsf{Verify}(\mathsf{vk}, \sigma, (\overrightarrow{\mathsf{vk}}||j)) \neq 1$.
       • Output “Special Abort” if $(\sigma, (\overrightarrow{\mathsf{vk}}||j)) \notin \mathcal{L}$.
       • Begin protocol $\pi^{\mathsf{Res}}$ amongst $\lambda^2$ parties $\mathsf{P}_1, \ldots, \mathsf{P}_{\lambda^2}$ playing the role of simulator Res.Sim
         on behalf of honest party $\mathsf{P}_j$ with identity vk. Res.Sim uses randomness $\mathsf{r}^{\mathsf{Res}}$.
       • Compute $\mathsf{msg}_1 = \mathsf{Res.Sim.NextMsg}_1(1^\lambda, \alpha; \mathsf{r}^{\mathsf{Res}})$, $\sigma_1 = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\ \mathsf{msg}_1||\ 1||j))$. Add
         $(\sigma_1, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$ to $\mathcal{L}$.
       • Output $(\mathsf{msg}_1, \sigma_1)$.
  5. If input = (“MPC”, “Round” $\mathsf{i}$, “Party” $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \overrightarrow{\mathsf{msg}}_{i-1}, \sigma_{i-1}, \mathsf{Trans}_{i-2}$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
           – $\mathsf{Verify}(\mathsf{vk}, \sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j)) \neq 1$.
       • Output “Special Abort” if $(\sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j))) \notin \mathcal{L}$.
       • If round number $i \leq \ell$, do:
           – Continue $\pi^{\mathsf{Res}}$. Compute $\mathsf{msg}_i = \mathsf{Res.Sim.NextMsg_i}\ (\mathsf{Trans}_{i-2}, \overrightarrow{\mathsf{msg}}_{i-1}; \mathsf{r}^{\mathsf{Res}})$. [a]
           – Compute $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}||\mathsf{msg}_i||\ i||j))$. Add $(\sigma_i, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}\ ||\mathsf{msg}_i||$
             $i||j))$ to $\mathcal{L}$.
           – Output $(\mathsf{msg}_i, \sigma_i)$.
       • If round number $i = (\ell + 1)$, do:
           – Compute $z = \mathsf{Res.Sim.Out}(\mathsf{Trans}_\ell, \overrightarrow{\mathsf{msg}}_\ell; \mathsf{r}^{\mathsf{Res}})$.
           – If $z = 0$, output $a$. Else, output $\perp$.
```

[a] Recall that Res.Sim is a straight line simulator and involves no rewinding. It extracts the adversary's input internally by a brute-force break.

Figure 4: Description of program C_2

```
Hardwired values: $(y, \alpha, \mathsf{sk}, \mathsf{vk}, \mathsf{r}^{\mathsf{Res}})$
Initialize list $\mathcal{L} = \emptyset$.
  1. Output $y$. Additionally, do the following.
  2. If input = ("Reveal Identity for MPC"): output vk.
  3. If input = ("MPC", "Begin", "Party" j, $\overrightarrow{\mathsf{vk}}$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
       • Output $\sigma = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||j))$. Add $(\sigma, (\overrightarrow{\mathsf{vk}}||j))$ to $\mathcal{L}$.
  4. If input = ("MPC", "Round" 1, "Party" j, $\overrightarrow{\mathsf{vk}}, \sigma$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
           – $\mathsf{Verify}(\mathsf{vk}, \sigma, (\overrightarrow{\mathsf{vk}}||j)) \neq 1$.
       • Output "Special Abort" if $(\sigma, (\overrightarrow{\mathsf{vk}}||j)) \notin \mathcal{L}$.
       • Begin protocol $\pi^{\mathsf{Res}}$ amongst $\lambda^2$ parties $\mathsf{P}_1, \ldots, \mathsf{P}_{\lambda^2}$ playing the role of simulator Res.Sim
         on behalf of honest party $\mathsf{P}_j$ with identity vk. Res.Sim uses randomness $\mathsf{r}^{\mathsf{Res}}$.
       • Compute $\mathsf{msg}_1 = \mathsf{Res.Sim.NextMsg}_1(1^\lambda, \alpha; \mathsf{r}^{\mathsf{Res}})$, $\sigma_1 = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\ \mathsf{msg}_1||\ 1||j))$. Add
         $(\sigma_1, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$ to $\mathcal{L}$.
       • Output $(\mathsf{msg}_1, \sigma_1)$.
  5. If input = ("MPC", "Round" i, "Party" j, $\overrightarrow{\mathsf{vk}}, \overrightarrow{\mathsf{msg}}_{i-1}, \sigma_{i-1}, \mathsf{Trans}_{i-2}$):
       • Output $\perp$ if any of the following happen:
           – $\mathsf{vk}^j \neq \mathsf{vk}$,
           – $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct,
           – $\mathsf{Verify}(\mathsf{vk}, \sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j)) \neq 1$.
       • Output "Special Abort" if $(\sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j))) \notin \mathcal{L}$.
       • If round number $i \leq \ell$, do:
           – Continue $\pi^{\mathsf{Res}}$. Compute $\mathsf{msg}_i = \mathsf{Res.Sim.NextMsg_i}\ (\mathsf{Trans}_{i-2}, \overrightarrow{\mathsf{msg}}_{i-1}; \mathsf{r}^{\mathsf{Res}})$.
           – Compute $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}||\mathsf{msg}_i||\ i||j))$. Add $(\sigma_i, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}\ ||\mathsf{msg}_i||$
             $i||j))$ to $\mathcal{L}$.
           – Output $(\mathsf{msg}_i, \sigma_i)$.
       • If round number $i = (\ell + 1)$, output $\perp$.
```

Figure 5: Description of program C_3

By a Markov argument, there exists a fixing of $a^* \in \{0,1\}^k$, C_{a^*} (and $h(a^*)$) such that

$$\left| \Pr[\mathcal{A}(\mathsf{Obf}(C_{a^*})) \oplus h(a^*) = 1] - \Pr[\mathsf{Obf.Sim}(1^{|C_{a^*}|}) \oplus h(a^*) = 1] \right| \geq \mu(\lambda)$$

```
Hardwired values: $(y, \alpha, \mathsf{sk}, \mathsf{vk}, \mathsf{r}^{\mathsf{Res}})$
Initialize list $\mathcal{L} = \emptyset$.
  1. Output $y$. Additionally, do the following.
  2. If input = ("Reveal Identity for MPC"): output vk.
  3. If input = ("MPC", "Begin", "Party" j, $\overrightarrow{\mathsf{vk}}$):
      • Output $\perp$ if any of the following happen:
          – $\mathsf{vk}^j \neq \mathsf{vk}$,
          – $\{\mathsf{vk}^k\}_{k\in[\lambda^2]}$ are not distinct,
      • Output $\sigma = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||j))$. Add $(\sigma, (\overrightarrow{\mathsf{vk}}||j))$ to $\mathcal{L}$.
  4. If input = ("MPC", "Round" 1, "Party" j, $\overrightarrow{\mathsf{vk}}, \sigma$):
      • Output $\perp$ if any of the following happen:
          – $\mathsf{vk}^j \neq \mathsf{vk}$,
          – $\{\mathsf{vk}^k\}_{k\in[\lambda^2]}$ are not distinct,
          – $\mathsf{Verify}(\mathsf{vk}, \sigma, (\overrightarrow{\mathsf{vk}}||j)) \neq 1$.
      • Output "Special Abort" if $(\sigma, (\overrightarrow{\mathsf{vk}}||j)) \notin \mathcal{L}$.
      • Begin protocol $\pi^{\mathsf{Res}}$ amongst $\lambda^2$ parties $\mathsf{P}_1, \ldots, \mathsf{P}_{\lambda^2}$ playing the role of honest party $\mathsf{P}_j$
        with identity vk using input $\alpha$, randomness $\mathsf{r}^{\mathsf{Res}}$ to evaluate functionality $f$.
      • Compute $\mathsf{msg}_1 = \pi^{\mathsf{Res}}.\mathsf{NextMsg}_1(1^\lambda, \alpha; \mathsf{r}^{\mathsf{Res}})$, $\sigma_1 = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\ \mathsf{msg}_1||\ 1||j))$. Add
        $(\sigma_1, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$ to $\mathcal{L}$.
      • Output $(\mathsf{msg}_1, \sigma_1)$.
  5. If input = ("MPC", "Round" i, "Party" j, $\overrightarrow{\mathsf{vk}}, \overrightarrow{\mathsf{msg}}_{i-1}, \sigma_{i-1}, \mathsf{Trans}_{i-2}$):
      • Output $\perp$ if any of the following happen:
          – $\mathsf{vk}^j \neq \mathsf{vk}$,
          – $\{\mathsf{vk}^k\}_{k\in[\lambda^2]}$ are not distinct,
          – $\mathsf{Verify}(\mathsf{vk}, \sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j)) \neq 1$.
      • Output "Special Abort" if $(\sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j))) \notin \mathcal{L}$.
      • If round number $i \leq \ell$, do:
          – Continue $\pi^{\mathsf{Res}}$. Compute $\mathsf{msg}_i = \pi^{\mathsf{Res}}.\mathsf{NextMsg}_i\ (\mathsf{Trans}_{i-2}, \overrightarrow{\mathsf{msg}}_{i-1}, \alpha; \mathsf{r}^{\mathsf{Res}})$.
          – Compute $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}||\mathsf{msg}_i||\ i||j))$. Add $(\sigma_i, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-1}\ ||\mathsf{msg}_i||$
            $i||j))$ to $\mathcal{L}$.
          – Output $(\mathsf{msg}_i, \sigma_i)$.
      • If round number $i = (\ell + 1)$, output $\perp$.
```

Figure 6: Description of program C_4

We assume w.l.o.g.[5] that $h(a^*) = 0$, then

$$\left|\Pr[\mathcal{A}(\mathsf{Obf}(C_{a^*})) = 1] - \Pr[\mathsf{Obf.Sim}(1^{|C_{a^*}|}) = 1]\right| \geq \mu(\lambda)$$

[5]Suppose $h(a^*) = 1$ for all a^* for which the equation holds, then

$$\left|\Pr[\mathcal{A}(\mathsf{Obf}(C_{a^*})) = 0] - \Pr[\mathsf{Obf.Sim}(1^{|C_{a^*}|}) = 0]\right| \geq \mu(\lambda)$$

which contradicts Definition 3.1. □

Lemma 5.9. *Assuming existential unforgeability of the signature scheme,*

$$\Big|\Pr[\mathsf{Hyb}_2 = 1] - \Pr[\mathsf{Hyb}_3 = 1]\Big| = \mathsf{negl}(\lambda)$$

Proof. The two hybrids are identical, unless $\mathsf{Obf.Sim}$ queries the oracle on:

1. Any input (“MPC”, “Round” 1, “Party” $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \sigma$) such that
   - $\mathsf{vk}^j = \mathsf{vk}$,
   - $\{\mathsf{vk}^k\}_{k\in[\lambda^2]}$ are all distinct,
   - $\mathsf{Verify}(\mathsf{vk}, \sigma, (\overrightarrow{\mathsf{vk}}||j)) = 1$, and
   - $(\sigma, (\overrightarrow{\mathsf{vk}}||j)) \notin \mathcal{L}$.

   (OR)
2. Any input (“MPC”, “Round” i, “Party” $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \overrightarrow{\mathsf{msg}}_{i-1}, \sigma_{i-1}, \mathsf{Trans}_{i-2}$) such that
   - $\mathsf{vk}^j = \mathsf{vk}$,
   - $\{\mathsf{vk}^k\}_{k\in[\lambda^2]}$ are all distinct,
   - $\mathsf{Verify}(\mathsf{vk}, \sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j)) = 1$, and
   - $(\sigma_{i-1}, (\overrightarrow{\mathsf{vk}}||\mathsf{Trans}_{i-2}||\mathsf{msg}_{i-1}||i-1||j))) \notin \mathcal{L}$.

But such a query contradicts existential unforgeability (under chosen message attack) of the signature scheme, and can therefore only be generated with probability $\mathsf{negl}(\lambda)$ (over the randomness of sampling vk). This completes the proof of the lemma. □

Lemma 5.10. *Assuming π^{Res} is an MPC protocol that is resettably secure against a malicious adversary that can corrupt up to all but one parties, admitting superpolynomial simulation,* $\Big|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]\Big| = \mathsf{negl}(\lambda)$

Proof. We show that if there exists a polynomial $p(\cdot)$ such that $\Big|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]\Big| \neq \mathsf{negl}(\lambda)$, then there exists a reduction $\mathcal{A}_{\mathsf{Res}}$ that contradicts security of the resettably secure MPC protocol π^{Res}. $\mathcal{A}_{\mathsf{Res}}$ does the following to respond to $\mathsf{Obf.Sim}$'s queries to the program that is has black-box access to.

- Compute parameters $(a, \alpha, y, \mathsf{SK}, \mathsf{vk})$ as done in Hyb_3. Initialize list $\mathcal{L} = \emptyset$.

which implies that

$$\Big|(1 - \Pr[\mathcal{A}(\mathsf{Obf}(C_{a^*})) = 1]) - (1 - \Pr[\mathsf{Obf.Sim}(1^{|C_{a^*}|}) = 1])\Big| \geq \mu(\lambda)$$

which implies

$$\Big|\Pr[\mathcal{A}(\mathsf{Obf}(C_{a^*})) = 1]) - \Pr[\mathsf{Obf.Sim}(1^{|C_{a^*}|}) = 1])\Big| \geq \mu(\lambda)$$

which contradicts Definition 3.1.

- When $\mathsf{Obf.Sim}$ queries $\mathcal{A}_{\mathsf{Res}}$ with an input of the form ("$\mathsf{Reveal\ Identity\ for\ MPC}$"), $\mathcal{A}_{\mathsf{Res}}$ responds with vk.

- When $\mathsf{Obf.Sim}$ queries $\mathcal{A}_{\mathsf{Res}}$ with an input of the form ("MPC", "Begin", "Party" $\mathsf{j}, \overrightarrow{\mathsf{vk}}$), $\mathcal{A}_{\mathsf{Res}}$ responds exactly as done by C_1 in Hyb_3: that is, output $\perp$ if $\mathsf{vk}^j \neq \mathsf{vk}$, (or) $\{\mathsf{vk}^k\}_{k \in [\lambda^2]}$ are not distinct. Send $\sigma = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||j))$ to $\mathsf{Obf.Sim}$ and add $(\sigma, (\overrightarrow{\mathsf{vk}}||j))$ to $\mathcal{L}$.

- When $\mathsf{Obf.Sim}$ queries $\mathcal{A}_{\mathsf{Res}}$ with an input of the form ("MPC", "Round" 1, "Party" $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \sigma$), $\mathcal{A}_{\mathsf{Res}}$ first computes the abort and "$\mathsf{Special\ Abort}$" steps as done by C_1 in Hyb_3. Then, $\mathcal{A}_{\mathsf{Res}}$ initiates an execution of protocol π^{Res} amongst λ^2 parties $\mathsf{P}_1, \ldots, \mathsf{P}_{\lambda^2}$ for functionality f defined in Figure 2 interacting with a challenger Chall. $\mathcal{A}_{\mathsf{Res}}$ plays the role of an adversary corrupting all parties except party P_j. It also sends the identity vk of party P_j to Chall and receives party P_j's round 1 message msg_1. Then, $\mathcal{A}_{\mathsf{Res}}$ computes $\sigma_1 = \mathsf{Sign}(\mathsf{sk}, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$, adds $(\sigma_1, (\overrightarrow{\mathsf{vk}}||\mathsf{msg}_1||1||j))$ to $\mathcal{L}$ and sends $(\mathsf{msg}_1, \sigma_1)$ to $\mathsf{Obf.Sim}$.

- When $\mathsf{Obf.Sim}$ queries $\mathcal{A}_{\mathsf{Res}}$ with an input of the form ("MPC", "Round" i, "Party" $\mathsf{j}, \overrightarrow{\mathsf{vk}}, \overrightarrow{\mathsf{msg}}_{i-1}$, $\sigma_{i-1}, \mathsf{Trans}_{i-2}$), $\mathcal{A}_{\mathsf{Res}}$ computes abort and "$\mathsf{Special\ Abort}$" steps as done by C_1 in Hyb_3. $\mathcal{A}_{\mathsf{Res}}$ continues an execution of protocol π^{Res} with challenger Chall. $\mathcal{A}_{\mathsf{Res}}$ sends round number i and $\overrightarrow{\mathsf{msg}}_{i-1} \setminus \mathsf{msg}^j_{i-1}$ to Chall - that is, the set of messages of all corrupt parties in round $(i-1)$. Chall responds back with P_j's round i message msg_i if $i \leq \ell$ and the output z if $i = (\ell + 1)$. Then, if $\imath \leq \ell$, $\mathcal{A}_{\mathsf{Res}}$ computes σ_i, adds it to $\mathcal{L}$ as in Hyb_3 and sends $(\mathsf{msg}_i, \sigma_i)$ to $\mathsf{Obf.Sim}$. If $i = (\ell + 1)$: if $z = 0$, $\mathcal{A}_{\mathsf{Res}}$ sends a to $\mathsf{Obf.Sim}$. Else, it sends $\perp$ to $\mathsf{Obf.Sim}$.

Observe that when the challenger honestly generates the messages of protocol π^{Res} on behalf of party P_j, the program that $\mathcal{A}_{\mathsf{Res}}$ provides $\mathsf{Obf.Sim}$ blackbox access to is exactly C_1 in Hyb_3 and when the challenger generates the messages of protocol π^{Res} on behalf of party P_j by running the simulator $\mathsf{Res.Sim}$ of the MPC protocol π^{Res}, the program that $\mathcal{A}_{\mathsf{Res}}$ provides $\mathsf{Obf.Sim}$ blackbox access to is exactly C_2 from Hyb_4. This completes the proof of the lemma. □

Lemma 5.11. $\Big|\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_5 = 1]\Big| = \mathsf{negl}(\lambda)$.

Proof. The two hybrids only differ when $\mathsf{Obf.Sim}$ makes a query with round set to $\ell + 1$, that results in $\mathsf{Res.Sim.Out}(\mathsf{Trans}_\ell, \overrightarrow{\mathsf{msg}}_\ell; \mathsf{r}^{\mathsf{Res}})$ being equal to 0. Note that the ideal functionality of the MPC protocol outputs 0 only if $\alpha_1 \oplus \ldots \alpha_{\lambda^2} = 0^{\sqrt{\lambda}}$. The probability (over random choice of honest party input α_j), that $\alpha_1 \oplus \ldots \alpha_{\lambda^2} = 0^{\sqrt{\lambda}}$ in any single instance of the MPC protocol is at most $2^{-\sqrt{\lambda}}$. Since the resettable MPC simulator can invoke this functionality at most $\mathsf{poly}(\lambda)$ times, the probability that the adversary makes a query where $\mathsf{Res.Sim.Out}(\mathsf{Trans}_\ell, \overrightarrow{\mathsf{msg}}_\ell; \mathsf{r}^{\mathsf{Res}}) = 0$, and therefore obtains an output that is not $\perp$, is $\mathsf{negl}(\lambda)$. □

Lemma 5.12. *Assuming π^{Res} is an MPC protocol that is resettably secure against a malicious adversary that can corrupt up to all but one parties, admitting superpolynomial simulation,* $\Big|\Pr[\mathsf{Hyb}_5 = 1] - \Pr[\mathsf{Hyb}_6 = 1]\Big| = \mathsf{negl}(\lambda)$.

Proof. This is identical to the proof of Lemma 5.10. □

Next, we prove the following lemma.

Lemma 5.13. *Assuming one-wayness of the function g, $\mathsf{Pr}[\mathsf{Hyb}_6 = 0] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$ where the probability is over the randomness of x and the adversary.*

Proof. Recall that the output of Hyb_6 is $s \oplus h(a)$, where s denotes the output of the adversary in Hyb_6. Thus it suffices to prove that $\mathsf{Pr}[s = h(a)] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$. Note that in Hyb_6, the program that $\mathsf{Obf.Sim}$ gets black-box access to is completely independent of the input a (and only has the output $g(a)$ hardwired). Therefore, if the lemma is not true, we can demonstrate a reduction $\mathcal{A}_g$ that predicts the hardcore bit of g with noticeable probability, which is a contradiction.

$\mathcal{A}_g$ obtains a string $g(a)$ externally from the challenger for a randomly chosen a, and sets $y = g(a)$. Using y, $\mathcal{A}_g$ samples remaining variables and answers oracle queries of $\mathsf{Obf.Sim}$ exactly as in Hyb_6. The output of $\mathcal{A}_g$ is identical to that of $\mathsf{Obf.Sim}$.

If the lemma is not true, then $\mathsf{Pr}[\mathcal{A}_g(g(a)) = h(a)] \geq \frac{1}{2} + \mu(\lambda)$ for some non-negligible $\mu(\cdot)$, which contradicts the fact that h is a Goldreich-Levin [GL89] hardcore bit of the (polynomially-secure) one-way function g. This leads to a contradiction, as desired. $\square$

It follows from lemmas 5.7 through 5.13 that $\mathsf{Pr}[\mathsf{Hyb}_6 = 0] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$ and

$$\Big| \Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_6 = 1] \Big| = \mathsf{negl}(\lambda)$$

where the latter is equivalent to

$$\Big| \Pr[\mathsf{Hyb}_0 = 0] - \Pr[\mathsf{Hyb}_6 = 0] \Big| = \mathsf{negl}(\lambda)$$

Therefore, we conclude that $\mathsf{Pr}[\mathsf{Hyb}_0 = 0] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$ which implies that

$$\Pr_{\mathsf{Hyb}_0}[\mathcal{A}(\hat{\mathsf{C}}) = h(a)] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

and since $h(a) = \mathsf{H}(x)$ in Hyb_0, we have

$$\Pr_{\mathsf{Hyb}_0}[\mathcal{A}(\hat{\mathsf{C}}) = \mathsf{H}(x)] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

as desired. This completes the proof.

5.3 Parallel Repetition Attack

We now describe the counterexample to the dream XOR lemma by setting $t = \lambda^2$. That is, we will construct a PPT adversary $\mathcal{A}$ that, given λ^2 samples of the outputs of the one way function $\mathcal{G}$, can compute the XOR of their respective hardcore bits with respect to predicate H with probability greater than or equal to $2^{-\sqrt{\lambda}}$ thus disproving Conjecture 5.3. Formally, we construct an adversary $\mathcal{A}$ such that : $\mathsf{Pr}[\mathcal{A}(\mathcal{G}(x_1), \ldots, \mathcal{G}(x_{\lambda^2})) = \mathsf{H}(x_1) \oplus \ldots \oplus \mathsf{H}(x_{\lambda^2})] \geq 2^{-\sqrt{\lambda}}$ where the probability is over the random choices of the values $(x_1, \ldots, x_{\lambda^2})$ and the randomness of the adversary.

Adversary's Strategy: Adversary $\mathcal{A}$, given $\{\mathcal{G}(x_i) = (y_i, \widehat{\mathsf{C}}_i)\}_{i\in[\lambda^2]}$, does:

1. Run an execution of the resettable MPC protocol π^{Res} amongst (λ^2) parties for functionality f (Figure 2) using obfuscated programs $\widehat{\mathsf{C}}_1, \ldots, \widehat{\mathsf{C}}_{\lambda^2}$. The protocol messages are forwarded appropriately to all the obfuscations.

2. Abort if any obfuscated program outputs $\perp$. Else, let a_i be the value output by $\widehat{\mathsf{C}}_i$ at the end of the protocol. For each $i \in [\lambda^2]$, compute $h(a_i)$.

3. Output $h(a_1) \oplus \ldots \oplus h(a_{\lambda^2})$.

Analysis: For each input $x_i = (a_i, b_i)$, recall that α_i is the first $\sqrt{\lambda}$ bits of $\mathsf{PRG}(b_i)$. For randomly chosen inputs $x_1, \ldots, x_{\lambda^2}$, $Pr[(\alpha_1 \oplus \ldots \oplus \alpha_{\lambda^2}) = 0^{\sqrt{\lambda}}] \geq 2^{-\sqrt{\lambda}}$. Therefore, by the correctness of the obfuscation scheme, the resettable MPC protocol π^{Res} and the signature scheme, it is easy to see that for randomly chosen inputs $x_1, \ldots, x_{\lambda^2}$, for every $j \in [\lambda^2]$, the adversary learns the pre-image a_i with probability $\geq 2^{-\sqrt{\lambda}}$. Thus, $\mathsf{Pr}[\mathcal{A}(\mathcal{G}(x_1), \ldots, \mathcal{G}(x_{\lambda^2})) = \mathsf{H}(x_1) \oplus \ldots \oplus \mathsf{H}(x_{\lambda^2})] \geq 2^{-\sqrt{\lambda}}$ and this completes the proof.

6 Counterexample via Differing-Inputs Obfuscation

In this section we give an alternate counterexample to the dream XOR lemma. In particular, we show how to use *public-coin differing-inputs obfuscation (PCdiO)* [BGI+01, ABG+13, BCP14, IPS15] (along with an injective one-way functions) to convert any counterexample to the dream version of direct-product hardness amplification into a counterexample for the dream XOR lemma. By combining this with the counterexample to dream direct-product hardness amplification of [DJMW12], which relies on a specialized hash function security property called "extended second-preimage resistance" (ESPR), we get a counterexample to the dream XOR lemma based on PCdiO, ESPR hashing, and injective one-way functions.

6.1 Counterexample to Direct-Product Amplification

We begin by recalling the result of [DJMW12], which relies on an ESPR hashing to get a counterexample to dream direct-product hardness amplification.

The direct-product (AKA parallel-repetition) lemma says that if inverting $f(x)$ on a random input x is weakly-hard (i.e., no polynomial time attacker succeeds with probability greater than $1/2$) then simultaneously inverting t independent copies $f(x_1), \ldots, f(x_t)$ for a sufficiently large $t = \mathsf{poly}(\lambda)$ is strongly-hard (i.e., no polynomial time attacker succeeds with better than negligible probability). The dream version of the direct-product lemma would strengthen the conclusion to requiring that no polynomial time attacker can succeed with probability better than 2^{-t}, or at least some particular negligible function such as $\mu(\lambda) = 2^{-\sqrt{\lambda}}$ as t gets sufficiently large.

The result of [DJMW12] gives a counterexample to the dream direct-product lemma. As a first step, they give a counterexample for a hard relation R where, given a random x it should be hard to find a value w such that $R(x, w) = 1$. They construct a relation that is weekly secure, meaning that no poly-time attacker can succeed with probability better than $1/2$. However, there is a poly-time attacker that can break t instances simultaneously (i.e., given random $x_1, \ldots, x_t$ find

$w_1, \dots, w_t$ such that $R(x_i, w_i) = 1$) with probability $\mu(\lambda) = 2^{-\sqrt{\lambda}}$, no matter how large t is. They then generically upgrade their counterexample for hard relations to one for one-way functions. It turns out that their counterexample for hard relations suffices for us. We give a broad overview of this counterexample based on ESPR hashing.

ESPR Hashing. Consider a fixed (un-keyed) hash function $h : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. Such a hash function is *extended second preimage resistant (ESPR)* if the following problem is hard. The adversary is given a random challenge $x \leftarrow \{0,1\}^n$. The adversary can choose any *Merkle path* of length $\ell \leq n$ that begins at x, meaning that we think of x as a leaf in a Merkle tree of height ℓ and the adversary specifies the position of x in the tree and provides the ℓ values corresponding to the siblings along the path from the leaf x to the root.[6] The path chosen by the adversary defines the value associated with the root of the tree y and the adversary's goal is to find some second preimage of y that differs from the one defined by the path. We restrict attention to hash functions h that are *weakly regular*, meaning that every output y has at least two preimages $(x_L, x_R) \neq (x'_L, x'_R)$ such that $h(x_L, x_R) = h(x'_L, x'_R)$.

Note that ESPR security is implied by collision-resistance, since an attack on EPSR requires finding a collision for y. However, we want ESPR security to hold for an un-keyed hash function, and collision resistance can only hold for keyed hash functions (with security against non-uniform attackers). The reason that we can hope for ESPR to hold for un-keyed hash functions is that the adversary is given a random leaf x as a challenge and hence cannot hard-code a collision for every such x. The work of [DJMW12] justified the ESPR property by showing that it holds in the random-oracle model with auxiliary input [Unr07, DGK17], which is a reasonable heuristic for modeling security properties satisfied by fixed (un-keyed) hash functions against non-uniform attackers. We refer the reader to [DJMW12] for the formal definition of ESPR.

Direct-Product Counterexample via ESPR. Consider the relation $R(x, w)$ that outputs 1 if w wins the ESPR game for challenge x. In other words w consists of a Merkle path that begins at x and a second-preiamge for the root associated with that Merkle path. It is easy to see that the direct-product does not amplify hardness of the relation R beyond 2^{-2n}. In particular, assume that an adversary is given t random values $x_1, \dots, x_t$. Then the adversary can construct a Merkle tree of depth $\log t$ whose leaves are $x_1, \dots, x_t$. This Merkle tree has some root y. The adversary can find a second-preimage of y by brute-force "guessing" with probability 2^{-2n}. If it succeeds then for every x_i it can come up with a witness w_i such that $R(x_i, w_i) = 1$ by using the Merkle path that starts at x_i to construct w_i. In other words, the adversary succeeds in breaking all t independent copies of the one-way relation in one shot by guessing a single hash preimage. This yields the following theorem.

Theorem 6.1 ([DJMW12]). *Assuming the existence of an ESPR-secure hash function, there exists a polynomial-time computable relation $R \subseteq \{\{0,1\}^n \times \{0,1\}^{p(n)}\}_{n \in \mathbb{N}}$ for some polynomial p such that the following two properties hold.*

- *Hardness: For all PPT $\mathcal{A}$, there is a negligible ν such that*

$$\Pr[R(x, \mathcal{A}(x)) = 1 \ : \ x \leftarrow \{0,1\}^n] \leq \nu(n).$$

[6]The adversary only needs to specify values along a single path. The adversary never needs to construct a complete tree (which may be of exponential size 2^ℓ) and the specified values may not even be consistent with *any* complete tree.

- *Non-Amplification: There exists a PPT adversary $\mathcal{A}^*$ such that for all polynomial $t = t(n)$ the following holds:*

$$\Pr\left[\bigwedge_{i\in[t]} R(x_i, w_i) = 1 \;:\; \begin{array}{c} x_1, \ldots, x_t \leftarrow \{0,1\}^n, \\ (w_1, \ldots, w_t) \leftarrow \mathcal{A}^*(x_1, \ldots, x_t) \end{array}\right] \geq 2^{-2n}.$$

Note that by scaling down n to $n = \sqrt{\lambda}/2$ for security parameter λ, we get a counterexample as claimed above, where security does not amplify beyond $2^{-\sqrt{\lambda}}$ no matter how many copies we take.

6.2 Counterexample to Dream XOR Lemma

We use the relation R from Theorem 6.1 to get a counterexample for the XOR lemma. A starting idea is to use a hard-core predicate of w as the predicate of the XOR lemma. One issue is that we only have a hard relation rather than a one-way relation and therefore we cannot even efficiently sample x with a valid witness w. Another issue is that there are many valid witnesses w for every x, and so the predicate is not even uniquely defined by x. We fix both of these issues. We will start with an injective one-way function $\hat{f}$ and define our counterexample one-way function f as follows. The input x for f consists of an input $\hat{x}$ for $\hat{f}$ and an instance $\tilde{x}$ of the relation R where $\tilde{x}$ is of length $n = \sqrt{\lambda}/2$. The output $y = f(x)$ will contain $\tilde{x}, \hat{y} = \hat{f}(\hat{x})$ along with an obfuscated circuit that takes as input a witness w and if $R(\tilde{x}, w) = 1$ it outputs $\hat{x}$ (the randomness for obfuscation is also made a part of the input x for f). We show that the function f is one-way and that, given the output y it is hard to find $\hat{x}$. Intuitively, this follows because it is hard to find a valid witness w for $\tilde{x}$ that will make the obfuscated circuit output anything useful, and therefore the obfuscated circuit is indistinguishable from one that does not contain $\hat{x}$; on the other hand the one-wayness of $\hat{f}$ says that it is hard to compute $\hat{x}$ from $\hat{y}$. We define a predicate $P(x)$ to be Goldreich-Levin hardcore bit of $\hat{x}$, and the above shows that $P(x)$ is a hard-core predicate of $f(x)$. On the other hand, it is easy to see that the dream XOR lemma does not hold for f, P. Given many values $y_i = f(x_i)$ for $i = 1, \ldots, t$, we can run the adversary $\mathcal{A}^*$ from Theorem 6.1 to find all the witnesses $w_1, \ldots, w_t$ for $\tilde{x}_1, \ldots, \tilde{x}_t$ with probability $2^{-2n} = 2^{-\sqrt{\lambda}}$. We then input these witnesses w_i to the respective obfuscated programs contained in y_i to recover $\hat{x}_i$, which allows us to recover all the hardcore-predicates $P(x_i)$ and therefore also $\bigoplus P(x_i)$.

While we do not know how to prove security via indistinguishability obfuscation (iO), we can do so via public-coin differing-inputs obfuscation (PCdiO) [ABG+13, BCP14, IPS15]. We first define PCdiO and then give a formal description and analysis of our counterexample.

Differing-Inputs Obfuscation. Differing-inputs obfuscation (diO) [BGI+01, ABG+13, BCP14] is a strengthening of indistinguishability obfuscation (iO) [BGI+01]. For iO we require that for any two circuits C_0, C_1 that are *functionally equivalent* the obfuscations of C_0, C_1 are computationally indistinguishable. For diO we consider a distribution over a tuple (C_0, C_1, aux) consisting of two circuits and some auxiliary information. Such a distribution is legal if given aux it is computationally hard to find a an inputs x such that $C_0(x) \neq C_1(x)$. The security of diO requires that for any legal distribution of (C_0, C_1, aux), the obfuscation of C_0 is indistinguishable from that of C_1 even given aux. Unfortunately, the work of [GGHW14] suggests that such a general form of diO is unlikely to be achievable, by construing a contrived distribution that is plausibly legal but is unobfuscatable. The work of [IPS15] refined the notion of diO to *public-coin diO (PCdiO)* by setting aux to include all of the random coins used to sample the tuple (C_0, C_1). This appears to preclude the counterexamples of [GGHW14] and is believed to be achievable in its full generality. In fact, all of the explicit candidate iO constructions can be conjectured to satisfy this stronger property. We recall the formal definition of PCdiO below.

Definition 6.2 (PCdiO [IPS15])**.** *A PPT algorithm* Obf *is public-coin differing-inputs obfuscation for circuits (PCdiO) if the following holds:*

- *Correctness: For all circuits C, all inputs x, and all values of $\lambda \in \mathbb{N}$ we have*

$$\Pr[\tilde{C}(x) = C(x) \ : \ \tilde{C} \leftarrow \mathsf{Obf}(1^\lambda, C)] = 1.$$

- *Security: We say that a PPT sampler* Sam *is* legal *if for $(C_0, C_1) \leftarrow \mathsf{Sam}(1^\lambda)$ we have $|C_0| = |C_1|$ and for all PPT $\mathcal{A}$ there is a negligible ν such that*

$$\Pr[C_0(x) \neq C_1(x) \ : \ (C_0, C_1) \leftarrow \mathsf{Sam}(1^\lambda; r), x \leftarrow \mathcal{A}(1^\lambda, r)] = \nu(\lambda).$$

  The PCdiO scheme is secure if for every legal sampler Sam *we have*

$$(r, \mathsf{Obf}(1^\lambda, C_0)) \approx_c (r, \mathsf{Obf}(1^\lambda, C_1))$$

  where $(C_0, C_1) \leftarrow \mathsf{Sam}(1^\lambda; r)$.

Counterexample. Let R be the relation from Theorem 6.1, let $\hat{f}$ be an injective one-way function, and let Obf be a PCdiO scheme. We assume that $\mathsf{Obf}(1^\lambda, C; r)$ uses randomness r of length $|r| = \lambda/3$; this is without loss of generality, since we can always use a pseuorandom generator to derive as many pseudorandom bits as needed from r. Let $n = \lfloor(\sqrt{\lambda} - 1)/2\rfloor$ and let $\ell = (\lambda - \lambda/3 - n)/2 = \Omega(\lambda)$.

We define a function $f(x)$ that takes as input $x \in \{0,1\}^\lambda$ and parses $x = (\hat{x}, \hat{r}, \tilde{x}, r) \in \{0,1\}^\ell \times \{0,1\}^\ell \times \{0,1\}^n \times \{0,1\}^{\lambda/3}$. It outputs $y = (\hat{y}, \hat{r}, \tilde{x}, \tilde{C})$ where $\hat{y} = \hat{f}(\hat{x})$, and $\tilde{C} = \mathsf{Obf}(1^\lambda, C_{\hat{x},\tilde{x}}; r)$ is computed by using r as the randomness for the obfuscation procedure that obfuscates the circuit $C_{\hat{x},\tilde{x}}$ defined as:

$$C_{\hat{x},\tilde{x}}(w) \ : \ \text{If } R(\tilde{x}, w) = 1 \text{ then output } \hat{x} \text{ else output } \perp. \tag{1}$$

For $x = (\hat{x}, \hat{r}, \tilde{x}, r)$, we also define the predicate $P(x) = \langle \hat{x}, \hat{r} \rangle$. This is the Goldreich-Levin hard-core bit of $\hat{x}$.

Theorem 6.3. *Assuming the security of PCdiO, ESPR hasing, and injective one-way functions, the construction of f, P above satisfies the following properties.*

- *Hardness: The function f is a one-way function and P is a hard-core predicate for it. In particular for all PPT $\mathcal{A}$ there is a negligible ν such that*

$$\Pr[\mathcal{A}(f(x)) = P(x) \ : \ x \leftarrow \{0,1\}^\lambda] \leq 1/2 + \nu(\lambda).$$

- *XOR Non-Amplification: There exists a PPT adversary $\mathcal{B}^*$ such that for all polynomial $t = t(\lambda)$:*

$$\Pr\left[\mathcal{B}^*(f(x_1), \ldots, f(x_t)) = \bigoplus_{i \in [t]} P(x_i)\right] \geq \frac{1}{2} + 2^{-\sqrt{\lambda}}.$$

Proof. For the first part of the theorem, we only need to show that given $f(x)$ it is hard to compute $\hat{x}$. Namely, for all PPT $\mathcal{A}$ there is a negligible ν such that

$$\Pr[\mathcal{A}(f(x)) = \hat{x} \ : \ x = (\hat{x}, \hat{r}, \tilde{x}, r) \leftarrow \{0,1\}^{\lambda}] \leq \nu(\lambda). \tag{2}$$

The fact that P is a hard-core predicate for f then follows by the Goldreich-Levin theorem [GL89]. Arguing by contradiction, assume (2) does not hold and there is some $\mathcal{A}$ whose success probability is non-negligible. Recall that $\mathcal{A}$ gets as input $f(x) = (\hat{y}, \hat{r}, \tilde{x}, \tilde{C})$. As a first step, we replace $\tilde{C} = \mathsf{Obf}(1^{\lambda}, C_{\hat{x},\tilde{x}}; r)$ by $\tilde{C}' = \mathsf{Obf}(1^{\lambda}, C'; r)$ where C' is a circuit that always outputs $\perp$, padded to be of the same size as $C_{\hat{x},\tilde{x}}$. We rely on PCdiO security to argue that this change is indistinguishable. Define $\mathsf{Sam}(1^{\lambda})$ to sample a uniform $\hat{x}, \tilde{x} \leftarrow \{0,1\}^{\ell} \times \{0,1\}^{n}$ and outputs the two circuits $(C_{\hat{x},\tilde{x}}, C')$. This is a legal sampler since, given $\hat{x}, \tilde{x}$, finding an input w on which $C_{\hat{x},\tilde{x}}(w) \neq C'(w)$ is equivalent to finding w on which $R(\tilde{x}, w) = 1$ which is hard by Theorem 6.1. Therefore PCdiO security implies that $\tilde{C}$ is indistinguishable from $\tilde{C}'$ even given $\hat{x}, \tilde{x}$. Therefore $\Pr[\mathcal{A}((\hat{y}, \hat{r}, \tilde{x}, \tilde{C}')) = \hat{x}]$ is non-negligible. But this means that we can construct an adversary $\mathcal{B}$ that breaks the one-wayness of $\hat{f}$ with non-neglgible success probability by defining $\mathcal{B}(\hat{y})$ to sample $\tilde{C}' \leftarrow \mathsf{Obf}(1^{\lambda}, C'), \hat{r} \leftarrow \{0,1\}^{\ell}, \tilde{x} \leftarrow \{0,1\}^{n}$ and output $\mathcal{A}(\hat{y}, \hat{r}, \tilde{x}, \tilde{C}')$. This yields a contradiction, proving the claim.

For the second part of the theorem, let $\mathcal{A}^*$ be the adversary on the direct-product of R from Theorem 6.1. Define $\mathcal{B}^*(y_1, \ldots, y_t)$ to parse $y_i = (\hat{y}_i, \hat{r}_i, \tilde{x}_i, \tilde{C}_i)$, run $(w_1, \ldots, w_t) \leftarrow \mathcal{A}^*(\tilde{x}_1, \ldots, \tilde{x}_t)$. If for some $i \in [t]$ we have $R(\tilde{x}_i, w_i) = 0$ then output a random bit. Else set $\hat{x}_i = \tilde{C}_i(w_i)$, $p_i = \langle \hat{x}_i, \hat{r}_i \rangle$ and output $\bigoplus_{i \in [t]} p_i$. We see that:

$$\begin{aligned}
\Pr[\mathcal{B}^* \text{ wins }] &= \Pr[\mathcal{A}^* \text{ wins }] + \frac{1}{2}(1 - \Pr[\Pr[\mathcal{A}^* \text{ wins }]]) \\
&= \frac{1}{2} + \frac{1}{2}\Pr[\mathcal{A}^* \text{ wins }] \\
&\geq \frac{1}{2} + 2^{-(2n+1)} \\
&\geq \frac{1}{2} + 2^{-\sqrt{\lambda}}.
\end{aligned}$$

□

Remarks. Note that the form of the counterexample above is somewhat stronger than the one from Section 5.3 since we do not need to fix t ahead of time. In other words, we have one fixed construction whose security does not amplify beyond $2^{-\sqrt{\lambda}}$ no matter how many copies t are taken, whereas previously the order of quantifiers was reversed and for every t we got a construction whose security does not amplify beyond $2^{-\sqrt{\lambda}}$.

Also, we note that instead of PCdiO we could have relied on the weaker notion of "public-coin extractable witness encryption", which is the natural public-coin analogue of extractable witness encryption [GKP+13]. Instead of obfuscating the circuit $C_{\hat{x},\tilde{x}}$ from equation (1) we would give a witness encryption of the message $\hat{x}$ using the statement $\tilde{x}$. The proof of security would otherwise be identical.

Lastly, note that we could have replaced $\sqrt{\lambda}$ with λ^{ϵ} for any $\epsilon > 0$ to get a counterexample where XOR of many predicates does not amplify security beyond $2^{-\lambda^{\epsilon}}$. Furthermore, if we assumed sub-exponential (resp. exponential) security of the injective one-way way functions and ESPR hash functions (as well as the PRG that expands small randomness r for the obfuscation, but this can be constructed from the correspondingly secure injective one-way function) then we could even push this to get a counter-example where XOR of many predicates does not amplify security beyond $2^{-(\log\lambda)^{O(1)}}$ (resp. $2^{-\log\lambda\log\log\lambda}$).

References

[ABG⁺13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013.

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In *EUROCRYPT*, 2019.

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, 2001.

[AJL⁺12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, 2012.

[AJL⁺19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In *CRYPTO*, 2019.

[AJN⁺16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In *CRYPTO*, 2016.

[AP16] Navid Alamati and Chris Peikert. Three's compromised too: Circular insecurity for any cycle length from (ring-)lwe. In *CRYPTO*, 2016.

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In *EUROCRYPT*, 2020.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73. Springer, Heidelberg, February 2014.

[BD18] Zvika Brakerski and Nico Döttling. Two-message statistically sender-private OT from LWE. In *TCC 2018*, 2018.

[BDGM20a] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate io from homomorphic encryption schemes. In *EUROCRYPT*, 2020.

[BDGM20b] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure LWE suffices. *IACR Cryptol. ePrint Arch.*, 2020.

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and uces: The case of computationally unpredictable sources. In *CRYPTO*, 2014.

[BGGL01] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. *FOCS*, 2001.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BGI08] Eli Biham, Yaron J. Goren, and Yuval Ishai. Basing weak public-key cryptography on strong one-way functions. In *TCC*, 2008.

[BGI$^+$17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In *ASIACRYPT*, 2017.

[BGJ$^+$17] Saikrishna Badrinarayanan, Vipul Goyal, Abhishek Jain, Dakshita Khurana, and Amit Sahai. Round optimal concurrent MPC via strong simulation. In *TCC*, 2017.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, 2009.

[BIN97] Mihir Bellare, Russell Impagliazzo, and Moni Naor. Does parallel repetition lower the error in computationally sound protocols? In *FOCS*, 1997.

[BL18] Nir Bitansky and Huijia Lin. One-message zero knowledge and non-malleable commitments. In *TCC*, 2018.

[BP13] Nir Bitansky and Omer Paneth. On the impossibility of approximate obfuscation and applications to resettable cryptography. In *STOC*, 2013.

[BP15] Nir Bitansky and Omer Paneth. On non-black-box simulation and the impossibility of approximate obfuscation. *SIAM J. Comput.*, 2015.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS*, 1993.

[BRT12] Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro. Multi-instance security and its application to password-based cryptography. In *CRYPTO*, 2012.

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

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, 2000.

[COP$^+$14] Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, Muthuramakrishnan Venkitasubramaniam, and Ivan Visconti. 4-round resettably-sound zero knowledge. In *TCC*, 2014.

[COPV13] Kai-Min Chung, Rafail Ostrovsky, Rafael Pass, and Ivan Visconti. Simultaneous resettability from one-way functions. In *FOCS*, 2013.

[CPS16] Kai-Min Chung, Rafael Pass, and Karn Seth. Non-black-box simulation from one-way functions and applications to resettable security. *SIAM J. Comput.*, 2016.

[DGI$^+$19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32. Springer, Heidelberg, August 2019.

[DGK17] Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 473–495. Springer, Heidelberg, April / May 2017.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO*, 2016.

[DJMW12] Yevgeniy Dodis, Abhishek Jain, Tal Moran, and Daniel Wichs. Counterexamples to hardness amplification beyond negligible. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 476–493. Springer, Heidelberg, March 2012.

[DQV$^+$21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct LWE sampling, random polynomials, and obfuscation. In Kobbi Nissim and Brent Waters, editors, *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, volume 13043 of *Lecture Notes in Computer Science*, pages 256–287. Springer, 2021.

[Fei91] Uriel Feige. On the success probability of the two provers in one-round proof systems. In *Structure in Complexity Theory Conference*. IEEE Computer Society, 1991.

[For89] Lance Fortnow. The complexity of perfect zero-knowledge. *Advances in Computing Research*, 1989.

[GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *TCC*, 2014.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In *CRYPTO*, 2014.

[GHRW14] Craig Gentry, Shai Halevi, Mariana Raykova, and Daniel Wichs. Outsourcing private RAM computation. In *FOCS*, 2014.

[GJLS21] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In *EUROCRYPT*, 2021.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553. Springer, Heidelberg, August 2013.

[GKW17a] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, 2017.

[GKW17b] Rishab Goyal, Venkata Koppula, and Brent Waters. Separating semantic and circular security for symmetric-key bit encryption from the learning with errors assumption. In *EUROCRYPT*, 2017.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, 1989.

[GM11] Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In *FOCS*, 2011.

[GNW11] Oded Goldreich, Noam Nisan, and Avi Wigderson. *On Yao's XOR-Lemma*. Springer Berlin Heidelberg, 2011.

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In *STOC*, 2021.

[GS09] Vipul Goyal and Amit Sahai. Resettably secure computation. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 54–71. Springer, Heidelberg, April 2009.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT*, 2018.

[GSV18] Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *FOCS*, 2018.

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In *ASIACRYPT*, 2000.

[HK12] Shai Halevi and Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. *J. Cryptol.*, 2012.

[HR14] Dennis Hofheinz and Andy Rupp. Standard versus selective opening security: Separation and equivalence results. In *TCC*, 2014.

[HRW16] Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In *TCC*, 2016.

[HY19] Justin Holmgren and Lisa Yang. The parallel repetition of non-signaling games: counterexamples and dichotomy. In Moses Charikar and Edith Cohen, editors, *51st Annual ACM Symposium on Theory of Computing*, pages 185–192. ACM Press, June 2019.

[Imp95] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *FOCS*, 1995.

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 668–697. Springer, Heidelberg, March 2015.

[JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over R to build io. In *EUROCRYPT*, 2019.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC*, 2021.

[JP11] Abhishek Jain and Krzysztof Pietrzak. Parallel repetition for leakage resilience amplification revisited. In *TCC*, 2011.

[KRW15] Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 378–400. Springer, Heidelberg, March 2015.

[KS17] Dakshita Khurana and Amit Sahai. How to achieve non-malleability in one or two rounds. In *FOCS*, 2017.

[KW16] Venkata Koppula and Brent Waters. Circular security separations for arbitrary length cycles from LWE. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 681–700. Springer, Heidelberg, August 2016.

[Lev85] Leonid A. Levin. One-way functions and pseudorandom generators. In *STOC*, 1985.

[LW10] Allison B. Lewko and Brent Waters. On the insecurity of parallel repetition for leakage resilience. In *51st Annual Symposium on Foundations of Computer Science*, pages 521–530. IEEE Computer Society Press, October 2010.

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

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

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In László Babai, editor, *STOC*, 2004.

[PW07] Krzysztof Pietrzak and Douglas Wikström. Parallel repetition of computationally sound protocols revisited. In *TCC*, 2007.

[Rot13] Ron Rothblum. On the circular security of bit-encryption. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 579–598. Springer, Heidelberg, March 2013.

[Sha20] Ronen Shaltiel. Is it possible to improve yao's XOR lemma using reductions that exploit the efficiency of their oracle? In *APPROX/RANDOM*, 2020.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.

[SV10] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 2010.

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In *CRYPTO*, 2007.

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In *EUROCRYPT*, 2021.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 600–611. IEEE Computer Society Press, October 2017.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, 1982.

A Resettable MPC: Definition

Consider n parties $P_1, \ldots, P_n$. An n-party functionality f is a (possibly randomized) mapping of n inputs to n outputs. A secure multi-party computation protocol π with security parameter λ for computing a functionality f is a protocol running in time $\mathsf{poly}(\lambda)$ with the goal of computing the output $f(x_1, \ldots, x_n)$.

The security of a protocol π (with respect to a functionality f) is defined by comparing the real-world execution of the protocol with an ideal-world evaluation of f by a trusted party. More concretely, it is required that for every adversary $\mathcal{A}$, which attacks the real execution of the protocol, there exist an adversary Sim, also referred to as a simulator, which can *achieve the same effect* in the ideal-world. We denote the set of all inputs by $\vec{x} = (x_1, \ldots, x_n)$. The adversary controls a set of parties and at any point during the execution of the protocol it can reset any of the honest parties. We shall consider computational security against parties which have been statically corrupted by the adversary. All honest parties have their random tape independently chosen but when an adversary resets a party, it reuses the same random tape (and the same input).

The real execution. In the real execution of the n-party protocol π for computing f in the presence of an adversary $\mathcal{A}$, the honest parties follow the instructions of π. The adversary $\mathcal{A}$, on input λ, outputs the identities of the honest parties, the indices I and identities in 2^λ of corrupted parties, the inputs of the corrupted parties, and an auxiliary input z. $\mathcal{A}$ sends all messages in place of corrupted parties and may follow an arbitrary polynomial-time strategy. The adversary can reset any honest party at any point of time during the execution of the protocol (and potentially even bring them out of sync). Recall that when an adversary resets a party, the reset party reuses the same random tape (and the same input).
The interaction of $\mathcal{A}$ with protocol π defines a random variable $\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda, \vec{x})$ whose value is determined by the coin tosses of the adversary and the honest players. This random variable

contains the output of the adversary (which may be an arbitrary function of its view) as well as the outputs of the uncorrupted parties at the end of the protocol. We let $\mathsf{REAL}_{\pi,\mathcal{A}(z),I}$ denote the distribution ensemble $\{\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda,\vec{x})\}_{\lambda\in\mathsf{N},\langle\vec{x},z\rangle\in\{0,1\}^*}$.

The ideal execution – security with abort. In the ideal world, there is a mutually trusted party which can aggregate the inputs provided to it by the various parties, perform the computation $f(\cdot)$ on their behalf and provide them their respective outputs. An ideal execution for a function f in the presence of an ideal-world adversary (simulator) Sim proceeds as follows:

- **Send inputs to the trusted party:** Honest parties send their inputs to the trusted party; but corrupted parties may decide to send modified inputs to the trusted party, as instructed by Sim. Let x'_i denote the value sent by P_i.
- **Trusted party sends output to the adversary:** The trusted party computes $f(x'_1,\ldots,x'_n) = (y_1,\ldots,y_n)$ and sends $\{y_i\}_{i\in I}$ to the adversary.
- **Adversary instructs trusted party to abort or continue:** This is formalized by having the adversary send either a continue or abort message to the trusted party. In the latter case, the trusted party sends to each uncorrupted party P_i its output value y_i. In the former case, the trusted party sends the special symbol $\perp$ to each uncorrupted party.
- **Resets:** The adversary can reset the ideal world at any point of time. When the adversary decides to reset the ideal world, it requests the trusted party to reset all honest parties and the trusted party sends a reset signal to all honest parties. At this point, the ideal world returns to the first stage where honest parties feed their inputs to the ideal functionality. The honest party inputs do not change between resets.
- **Outputs:** Sim outputs an arbitrary function of its view, and the honest parties output the values obtained from the trusted party.

Sim's interaction with the trusted party defines a random variable $\mathsf{IDEAL}_{f_\perp,\mathcal{A}(z),I}(\lambda,\vec{x})$ that denotes the distribution ensemble $\{\mathsf{IDEAL}_{f_\perp,\mathsf{Sim}(z),I}(\lambda,\vec{x})\}_{\lambda\in\mathsf{N},\langle\vec{x},z\rangle\in\{0,1\}^*}$ where the subscript "$\perp$" indicates that the adversary can abort computation of f. Having defined the real and the ideal worlds, we now proceed to define our notion of security.

Definition A.1. *[Resettable MPC with Straight-Line, Black-Box Superpolynomial Simulation] Let λ be the security parameter. Let f be an n-party randomized functionality, and π be an n-party protocol for $n \in \mathsf{N}$.*

We say that π computes f with resettable straight-line, black-box superpolynomial simulation in the presence of malicious adversaries if for every PPT adversary $\mathcal{A}$ there exists a super-polynomial time simulator Sim that interacts with the adversary via straight-line black-box queries, such that for any $I \subset [n]$:

$$|\Pr[\mathsf{REAL}_{\pi,\mathcal{A}(z),I}(\lambda,\vec{x}) = 1] - \Pr[\mathsf{IDEAL}_{f_\perp,\mathsf{Sim}(z),I}(\lambda,\vec{x}) = 1]| = \mathsf{negl}(\lambda)$$

where $\vec{x} = \{x_i\}_{i\in[n]} \in \{0,1\}^*$ and $z \in \{0,1\}^*$.

The simulator Sim is allowed to reset the functionality in the ideal world several times and the number of times the ideal functionality is reset by the simulator could possibly be more than the number of resets performed by the adversary in the real world, though this number must be a polynomial in the security parameter.

A.1 Security Against Semi-Malicious Adversaries

We take this definition verbatim from [AJL$^+$12]. A semi-malicious adversary is modeled as an interactive Turing machine (ITM) which, in addition to the standard tapes, has a special witness tape. In each round of the protocol, whenever the adversary produces a new protocol message msg on behalf of some party P_k, it must also write to its special witness tape some pair (x, r) of input x and randomness r that explains its behavior. More specifically, all of the protocol messages sent by the adversary on behalf of P_k up to that point, including the new message m, must exactly match the honest protocol specification for P_k when executed with input x and randomness r. Also, we assume that the attacker is rushing and hence may choose the message m and the witness (x, r) in each round adaptively, after seeing the protocol messages of the honest parties in that round (and all prior rounds). The adversary may also choose to abort the execution on behalf of P_k in any step of the interaction.

B Resettable MPC: Proof

B.1 Resettable Semi-Malicious MPC: Proof

In this section, we formally prove Theorem 4.2.

Simulator Description:

Consider a semi-malicious adversary $\mathcal{A}$ that corrupts t parties where $t < n$. The strategy of the simulator Sim against the adversary $\mathcal{A}$ is described below:

1. **Round 1:**
   For each honest party P_i, Sim does the following:
   - Compute and send $\mathsf{msg}_{1,i} \leftarrow \mathsf{Commit}(0^{|k|+|x|})$.
2. **Rounds $2 \dots \ell$:**
   Let the round number be j. For each honest party P_i, including on each reset execution, Sim does the following:
   - Let Trans_{j-1} denote the transcript of the underlying protocol π at the end of round $(j-1)$, excluding the messages sent in round 1.
   - Compute and send $\mathsf{msg}_{j,i} \leftarrow \mathcal{S}_{j-1}(\mathsf{Trans}_{j-1}, \mathsf{x}^*, \mathsf{r}^*_{j-1})$ where $\mathsf{x}^*, \mathsf{r}^*_{j-1}$ denote the vector of malicious party inputs, and the randomness of the malicious parties up to round $(j-1)$ in the underlying protocol π. If $j = 2$, we set $(\mathsf{x}^*_{j-1}, \mathsf{r}^*_{j-1} = \perp)$.
   - Note that each time honest party P_i is reset to the beginning of a particular round j, if the adversary sent the same message as some previous execution, $\mathcal{S}_j$ uses the same randomness as in that execution and hence the message $\mathsf{msg}_{j,i}$ doesn't change.

Remarks: Here, note that our simulator is a PPT algorithm. Only in the next subsection when we consider malicious adversaries, we build simulators that run in super-polynomial time.

Hybrids: We now show that the simulation strategy described above is successful against all resetting semi-malicious PPT adversaries. That is, the view of the adversary along with the output of the honest parties is computationally indistinguishable in the real and ideal worlds. We will show this via a series of computationally indistinguishable hybrids where the first hybrid Hyb_1 corresponds to the real world and the last hybrid Hyb_3 corresponds to the ideal world.

Before describing this sequence, we note that even under $t < n$ corruptions, since the adversary is guaranteed to be semi-malicious, all the protocol messages sent by a semi-malicious adversary will be a **deterministic function** of their first round message. Therefore, any semi-malicious adversary, upon resetting honest parties to the middle of the j^{th} round for $j \geq 2$, sends a message that is a deterministic function of the first round message, and therefore repeatedly sends the exact same message.

1. Hyb_1**:** This hybrid outputs the view of the adversary when interacting with honest parties.
2. Hyb_2**:** In this hybrid, the challenger computers $\mathsf{msg}_{1,i} \leftarrow \mathsf{Commit}(0^{|k|+|\mathsf{x}|})$ for every honest party P_i.
3. Hyb_3**:** In this hybrid, for every honest party P_i, the first round message is computed identically to Hyb_2, but the simulator computes the messages for every other round of π using uniform randomness instead of the PRF output. That is, for $j \geq 2$, it computes and sends $\mathsf{msg}_{j,i} \leftarrow \pi.\mathsf{NextMsg_j}(\mathsf{x}_i, \mathsf{Trans}_{j-1}; \mathsf{r}_{j,i})$ where the randomness $\mathsf{r}_{j,i}$ is picked uniformly at random. When the adversary resets and sends a new round 1 message, then the simulator does the following:
   - If the adversary sends a message that is the same as one sent in a previous execution, use randomness that is identical to the one used in that specific execution.
   - If the adversary sends a message that is different from one sent in a previous execution for the same transcript prefix, this adversary is *not explainable*, and therefore no guarantees are required against such an adversary: it is safe to answer arbitrarily.
4. Hyb_4**:** In this hybrid, for every honest party P_i, for every round j with $j \in [\ell]$, including on each reset to the beginning of round 2, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the message of the underlying protocol π using the simulator $\mathcal{S}$ for the semi-malicious MPC protocol. If the adversary sends the same message in round 1 as a previous execution, then $\mathsf{Sim}_{\mathsf{Hyb}}$ continues this execution identically to the previous execution. This experiment corresponds to the ideal world.

Indistinguishability of Hybrids: We now show that every pair of successive hybrids is computationally indistinguishable.

Claim B.1. *Assuming* Commit *is a computationally hiding commitment,* Hyb_1 *is computationally indistinguishable from* Hyb_2.

Proof. The only difference between the two hybrids is that in Hyb_1, honest parties to commit to their actual inputs and to a PRF key in round 1, whereas in Hyb_2, the challenger sends a commitment to 0 in round 1. Any adversary that distinguishes these two hybrids breaks the hiding property of the commitment scheme. $\square$

Claim B.2. *Assuming that* PRF *is a secure pseudorandom function family,* Hyb_2 *is computationally indistinguishable from* Hyb_3.

Proof. The only difference between the two hybrids is that in Hyb_2, for each honest party P_i, for every round j with $j \in [\ell]$, including on each reset, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the randomness for the underlying protocol π as the output of the PRF on the transcript so far, while in Hyb_3, a uniformly random string is picked whenever the adversary resets and sends a different message.

That is, in Hyb_2, for any round j, the message sent by honest party P_i is $\mathsf{msg}_{j,i} \leftarrow \pi.\mathsf{NextMsg}_\mathsf{j}(\mathsf{x}_i, \mathsf{Trans}_{j-1}; \mathsf{r}_{j,i})$ where the randomness $\mathsf{r}_{j,i} = \mathsf{PRF}(\mathsf{k}_i, \mathsf{Trans}_{j-1})$ and Trans_{j-1} is the transcript of the protocol π^{RSM} up to round j. On the other hand, in Hyb_2, $\mathsf{r}_{j,i}$ is a uniformly random string picked afresh on each reset (whenever the adversary sends a message that was not sent during a previous reset). Observe that we do not use the PRF key k_i anywhere in the protocol except to compute this randomness. Therefore, any adversary $\mathcal{A}$ that can distinguish between these two hybrids implies an adversary $\mathcal{A}_{\mathsf{PRF}}$ that breaks the security of the pseudorandom function PRF, which is a contradiction. □

Claim B.3. *Assuming semi-malicious security of the protocol π against adversaries that can corrupt up to all but one parties,* Hyb_3 *is computationally indistinguishable from* Hyb_4.

Proof. The difference between the two hybrids is that in Hyb_3, on behalf of the honest parties, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the messages of protocol π using the honest parties' actual inputs (and fresh randomness on each reset), whereas in Hyb_3, they are computed by running the simulator $\mathcal{S}$ for the underlying semi-malicious secure protocol π.

Let s denote the number of sessions opened by $\mathcal{A}$. We consider a sequence of hybrid experiments $\mathsf{Hyb}_{2,0}, \ldots, \mathsf{Hyb}_{2,s}$, with $\mathsf{Hyb}_{2,1} \triangleq \mathsf{Hyb}_2$, and $\mathsf{Hyb}_{2,s} \triangleq \mathsf{Hyb}_3$. For every $i \in [s]$, $\mathsf{Hyb}_{2,i}$ is defined as follows:

- For each session $i' \in [s]$ that starts at round 2 with prefix transcript $\tau_{i'}$, do the following.
  - If $i' \leq i$, execute the simulator $\mathcal{S}_j$ for semi-malicious MPC to compute messages on behalf of honest parties.[7]
  - If $i' > i$, use honest party strategy to compute messages on behalf of honest parties.

If there exists $i \in [\ell]$ and an adversary $\mathcal{A}$ that can distinguish between Hyb_{i-1} and Hyb_i, we will use $\mathcal{A}$ to design an algorithm $\mathcal{A}_\pi$ that can break the semi-malicious security of protocol π.

- $\mathcal{A}_\pi$ interacts with a challenger $\mathcal{C}$ to break the security of protocol π.
- $\mathcal{A}_\pi$ corrupts the same parties as $\mathcal{A}$ in its interaction with $\mathcal{C}$.
- $\mathcal{A}_\pi$ samples the first round messages for all honest parties as commitments to 0.
- For all sessions $(1, \ldots i-1)$, $\mathcal{A}_\pi$ follows the strategy in Hyb_3.
- For all sessions $(i+1, \ldots, s)$, $\mathcal{A}_\pi$ follows the strategy in Hyb_4.
- For the i^{th} session, for rounds $(2, \ldots, \ell)$, $\mathcal{A}_\pi$ obtains the messages for all honest parties, that are either sampled by $\mathcal{C}$ honestly according to π, or sampled using simulator $\mathcal{S}$ for the i^{th} session, and forwards them to $\mathcal{A}$ on behalf of honest parties.

[7]Note that the adversary is semi-malicious after round 2.

Now, observe that when the challenger $\mathcal{C}$ computes the messages on behalf of the honest parties using the honest parties' actual inputs, the interaction between $\mathcal{A}_\pi$ and $\mathcal{A}$ corresponds to $\mathsf{Hyb}_{2,i-1}$. Similarly, when $\mathcal{C}$ computes the messages on behalf of the honest parties by running the simulator $\mathcal{S}$, the interaction between $\mathcal{A}_\pi$ and $\mathcal{A}$ corresponds to $\mathsf{Hyb}_{2,i}$. Therefore, if there exists a distinguisher $\mathcal{D}$ that distinguishes between the two hybrids with non-negligible probability, $\mathcal{A}_\pi$ can use the same distinguisher to break the semi-malicious security of protocol π, which is a contradiction. $\square$

B.2 Resettable Malicious Secure MPC: Proof

Several parts of this section are copied verbatim from [BGJ+17].

Let f be any functionality. Consider n parties $\mathsf{P}_1, \ldots, \mathsf{P}_n$ with inputs $\mathsf{x}_1, \ldots, \mathsf{x}_n$ respectively who wish to compute f on their joint inputs by running a resettably secure MPC protocol. Let π^{SM} be any three round protocol for the above task that is secure against adversaries that can be completely malicious in the first round, but promise to be semi-malicious in the next two rounds, and can corrupt upto $(n-1)$ parties. In this section, we show how to generically transform π^{SM} into a three round protocol π that is resettably secure against malicious adversaries that can corrupt upto $(n-1)$ parties, using super-polynomial simulation. Formally, we prove the following theorem:

B.2.1 Construction

We first list some notation and the primitives used before describing the construction.

Notation.

- λ denotes the security parameter.
- $\mathsf{SPSS.ZK} = (\mathsf{ZK}_1, \mathsf{ZK}_2, \mathsf{ZK}_3)$ is a two message zero knowledge argument with super polynomial strong simulation (SPSS-ZK). The zero knowledge property holds against all adversaries running in time T_{ZK}. Let $\mathsf{Sim}^{\mathsf{ZK}}$ denote the simulator that produces simulated ZK proofs and let $\mathsf{T}_{\mathsf{ZK}}^{\mathsf{Sim}}$ denote its running time. [KS17] give a construction of an $\mathsf{SPSS.ZK}$ scheme satisfying these properties that can be based on sub-exponential DDH.
- $\mathsf{NMCom} = (\mathsf{NMCom}_1^R, \mathsf{NMCom}_2^S)$ is a two message concurrent non-malleable commitment scheme with respect to commitment in the simultaneous message model. Here, NMCom_1^R denote the first message of the receiver while NMCom_2^S denotes the second message from the sender. It is secure against all adversaries running in time $\mathsf{T}_{\mathsf{Com}}^{\mathsf{Sec}}$, but can be broken by adversaries running in time $\mathsf{T}_{\mathsf{Com}}^{\mathsf{Brk}}$. Let $\mathsf{Ext.Com}$ denote a brute force algorithm running in time $\mathsf{T}_{\mathsf{Com}}^{\mathsf{Brk}}$ that can break the commitment scheme. [KS17] give a construction of an NMCom scheme satisfying these properties that can be based on sub-exponential DDH.

  The NMCom we use is tagged. In the authenticated channels setting, the tag of each user performing a non-malleable commitment can just be its identity. In the general setting, in the first round, each party can choose a strong digital signature verification key VK and signing key, and then sign all its messages using this signature scheme for every message sent in the protocol. This VK is then used as the tag for all non-malleable commitments. This ensures that every adversarial party must choose a tag that is different than any tags chosen by honest parties, otherwise the adversary will not be able to sign any of its messages by the existential unforgeability property of the signature scheme. This is precisely the property that is assumed when applying NMCom. For ease of notation, we suppress writing the tags explicitly in our protocols below.

- π^{RSM} is a sub-exponentially secure 3 round MPC protocol that is resettably secure against adversaries running in time T_{SM} that can behave maliciously in the first round and semi-maliciously in the next two rounds. Let $(\pi^{\mathsf{RSM}}.\mathsf{NextMsg}_1, \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_2, \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_3)$ denote the algorithms used by any party to compute the messages in each of the three rounds and $\pi^{\mathsf{RSM}}.\mathsf{OUT}$ denotes the algorithm to compute the final output. Further, let's assume that π^{RSM} runs over a broadcast channel. Let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ denote the straight line simulator for this protocol - that is, $\mathcal{S}_i$ is the simulator's algorithm to compute the i^{th} round messages. Also, we make the following assumptions about the protocol structure, that is satisfied by the instantiations:
  1. $\mathcal{S}_1$ and $\mathcal{S}_2$ run without any input other than the protocol transcript so far - in particular, they don't need the input, randomness and output of the malicious parties. For $\mathcal{S}_1$, this must necessarily be true since the first round of π^{RSM} is secure against malicious adversaries. We make the assumption only on $\mathcal{S}_2$.
  2. The algorithm $\pi^{\mathsf{RSM}}.\mathsf{NextMsg}_3$ doesn't require any new input or randomness that was not already used in the algorithms $\pi^{\mathsf{RSM}}.\mathsf{NextMsg}_1$, $\pi^{\mathsf{RSM}}.\mathsf{NextMsg}_2$. Looking ahead, this is used in our security proof when we want to invoke the simulator of this protocol π^{RSM}, we need to be sure that we have fed the correct input and randomness to the simulator. This is true for all instantiantions we consider, where the semi-malicious simulator requires only the secret keys of corrupted parties (that are fixed in the second round) apart from the protocol transcript.
- In order to realize our protocol, we require that $\mathsf{poly}(\lambda) < \mathsf{T}^{\mathsf{Sim}}_{\mathsf{ZK}} < \mathsf{T}^{\mathsf{Sec}}_{\mathsf{Com}} < \mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}} < \mathsf{T}_{\mathsf{ZK}}, \mathsf{T}_{\mathsf{SM}}$.
- We assume broadcast channels. Further, let's assume that every party has an associated identity id. For any session sid, each party generates its non-malleable commitment using the tag $(\mathsf{id}||\mathsf{sid})$.

NP Languages. In our construction, we use proofs for the following NP languages.

NP language L is characterized by the following relation R.
Statement : $\mathsf{st} = (\mathsf{c}_1, \mathsf{c}_2, \mathsf{msg}_1, \mathsf{msg}_2, \tau)$
Witness : $\mathsf{w} = (\mathsf{inp}, \mathsf{r}, \mathsf{r_c})$
$R(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

- $\mathsf{c}_2 = \mathsf{NMCom}^S_2(\mathsf{inp}, \mathsf{r}, \mathsf{c}_1; \mathsf{r_c})$ AND
- $\mathsf{msg}_1 = \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_1(\mathsf{inp}; \mathsf{r})$ AND
- $\mathsf{msg}_2 = \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_2(\mathsf{inp}, \tau; \mathsf{r})$

That is, $(\mathsf{c}_1, \mathsf{c}_2)$ form a non-malleable commitment of $(\mathsf{inp}, \mathsf{r})$ such that msg_2 is the second round message using input inp, randomness r when running π^{RSM}, where the protocol transcript so far is τ.

NP language L_1 is characterized by the following relation R_1.
Statement : $\mathsf{st} = (\mathsf{c}_1, \mathsf{c}_2, \mathsf{msg}_3, \tau)$
Witness : $\mathsf{w} = (\mathsf{inp}, \mathsf{r}, \mathsf{r}_\mathsf{c})$
$R(\mathsf{st}, \mathsf{w}) = 1$ if and only if :

- $\mathsf{c}_2 = \mathsf{NMCom}_2^S(\mathsf{inp}, \mathsf{r}, \mathsf{c}_1; \mathsf{r}_\mathsf{c})$ AND
- $\mathsf{msg}_3 = \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_3(\mathsf{inp}, \tau; \mathsf{r})$

$(\mathsf{c}_1, \mathsf{c}_2)$ form a non-malleable commitment of $(\mathsf{inp}, \mathsf{r})$ such that msg_3 is the third round message using input inp, randomness r when running π^{RSM}, where the protocol transcript so far is τ.

Construction. The protocol is described in Figure 7.

B.2.2 Security Proof

In this section, we formally prove security of the above construction.

Consider an adversary $\mathcal{A}$ who corrupts $t < n$ parties. For each party P_i, let's say that the size of input and randomness used in protocol π^{RSM} is $p(\lambda)$ for some polynomial p. That is, $|(\mathsf{x}_i, \mathsf{r}_i)| = p(\lambda)$. The strategy of the simulator Sim against a malicious adversary $\mathcal{A}$ is described in *Figure* 8.

In the simulation, we crucially use the two assumptions about the protocol structure. The first one is easy to notice since the simulator Sim has to run the semi-malicious to produce the first and second messages before it has extracted the adversary's input and randomness. For the second assumption, observe that to run the simulator algorithm $\mathcal{S}_3$, Sim has to feed it the entire input and randomness of the adversary and so these have to be fixed to by the end of the second round.

We now show that the simulation strategy is successful against all malicious PPT adversaries. That is, the view of the adversary along with the output of the honest parties is computationally indistinguishable in the real and ideal worlds. We will show this via a series of computationally indistinguishable hybrids where the first hybrid Hyb_1 corresponds to the real world and the last hybrid Hyb_6 corresponds to the ideal world.

1. Hyb_1**:** In this hybrid, consider a simulator $\mathsf{Sim}_{\mathsf{Hyb}}$ that plays the role of the honest parties including on each reset execution. $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in polynomial time.

2. Hyb_2**:** In this hybrid, the simulator $\mathsf{Sim}_{\mathsf{Hyb}}$ also runs the "Query to Ideal Functionality" phase and the "Special Abort" phase in step3 and 5 in Figure 8 including on each reset execution. $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$.

3. Hyb_3**:** This hybrid is identical to the previous hybrid except that in Rounds 2 and 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ now computes simulated SPSSZK arguments as done in Figure 8 including on each reset execution. Once again, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$.

4. Hyb_4**:** This hybrid is identical to the previous hybrid except that $\mathsf{Sim}_{\mathsf{Hyb}}$ now computes all the $(\mathsf{c}^j_{2,i})$ as non-malleable commitments of $0^{p(\lambda)}$ as done in Round 2 in Figure 8 including on each reset execution. Once again, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$.

Inputs: Each party P_i has input x_i and uses randomness r_i to compute the message in each round of the protocol π^{RSM}. We now describe the messages sent by party P_i. We will use superscripts to denote the intended recipient of the message if it is not meant to be used by all parties.

1. **Round 1:**
   P_i does the following:
   - Compute $\mathsf{msg}_{1,i} \leftarrow \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_1(\mathsf{x}_i;\mathsf{r}_i)$.
   - For each $j \in [n]$ with $j \neq i$, compute:
     - $\mathsf{c}^j_{1,i} \leftarrow \mathsf{NMCom}^R_1(1^\lambda)$.
     - $(\mathsf{ver}^j_{1,i}, \mathsf{zkst}^j_{1,i}) \leftarrow \mathsf{ZK}_1(1^\lambda)$ and $(\mathsf{ver}^j_{2,i}, \mathsf{zkst}^j_{2,i}) \leftarrow \mathsf{ZK}_1(1^\lambda)$.
   - Send $(\mathsf{msg}_{1,i}, \mathsf{c}^j_{1,i}, \mathsf{ver}^j_{1,i}, \mathsf{ver}^j_{2,i})$ for all j.
2. **Round 2:**
   Let τ_1 denote the protocol transcript after round 1. P_i does the following:
   - Compute $\mathsf{msg}_{2,i} \leftarrow \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_2(\mathsf{x}_i, \tau_1;\mathsf{r}_i)$.
   - For each $j \in [n]$ with $j \neq i$, compute:
     - $\mathsf{c}^j_{2,i} \leftarrow \mathsf{NMCom}^S_2(\mathsf{x}_i, \mathsf{r}_i, \mathsf{c}^i_{1,j}; \mathsf{r}^j_{\mathsf{c},i})$ using the same random string $\mathsf{r}^j_{\mathsf{c},i}$.
     - $\mathsf{prove}^j_{2,i} \leftarrow \mathsf{ZK}_2(\mathsf{ver}^i_{1,j}, \mathsf{st}^j_{2,i}, \mathsf{w}^j_{2,i})$ for the statement $\mathsf{st}^j_{2,i} = (\mathsf{c}^i_{1,j}, \mathsf{c}^j_{2,i}$, $\mathsf{msg}_{1,i}$, $\mathsf{msg}_{2,i}$, $\tau_1) \in L$ using witness $\mathsf{w}^j_{2,i} = (\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}^j_{\mathsf{c},i})$.
   - Send $(\mathsf{msg}_{2,i}, \mathsf{c}^j_{2,i}, \mathsf{prove}^j_{2,i})$ for all j.
3. **Round 3:**
   Let τ_2 denote the protocol transcript after round 2. P_i does the following:
   - Compute $\mathsf{msg}_{3,i} \leftarrow \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_3(\mathsf{x}_i, \tau_2;\mathsf{r}_i)$.
   - For each $j \in [n]$ with $j \neq i$, do:
     - Abort if $\mathsf{ZK}_3(\mathsf{zkst}^j_{1,i}, \mathsf{st}^i_{2,j}) \neq 1$ where $\mathsf{st}^i_{2,j} = (\mathsf{c}^j_{1,i}, \mathsf{c}^i_{2,j}, \mathsf{msg}_{1,j}, \mathsf{msg}_{2,j}, \tau_1)$.
     - $\mathsf{prove}^j_{3,i} \leftarrow \mathsf{ZK}_2(\mathsf{ver}^i_{2,j}, \mathsf{st}^j_{3,i}, \mathsf{w}^j_{3,i})$ for the statement $\mathsf{st}^j_{3,i} = (\mathsf{c}^i_{1,j}, \mathsf{c}^j_{2,i}$, $\mathsf{msg}_{3,i}, \tau_2) \in L_1$ using witness $\mathsf{w}^j_{3,i} = (\mathsf{x}_i, \mathsf{r}_i, \mathsf{r}^j_{\mathsf{c},i})$.
   - Send $(\mathsf{msg}_{3,i}, \mathsf{prove}^j_{3,i})$ for all j.
4. **Output Computation:**
   Let τ_3 denote the protocol transcript after round 3. P_i does the following:
   - For each $j \in [n]$ with $j \neq i$, do:
     - Abort if $\mathsf{ZK}_3(\mathsf{zkst}^j_{2,i}, \mathsf{st}^i_{3,j}) \neq 1$ where $\mathsf{st}^i_{3,j} = (\mathsf{c}^j_{1,i}, \mathsf{c}^i_{2,j}, \mathsf{msg}_{3,j}, \tau_2)$.
   - Compute output $\mathsf{y}_i \leftarrow \pi^{\mathsf{RSM}}.\mathsf{OUT}(\mathsf{x}_i, \tau_3;\mathsf{r}_i)$.

Figure 7: 3 round MPC Protocol π for functionality f.

5. Hyb_5: This hybrid is identical to the previous hybrid except that in Round 3, $\mathsf{Sim}_{\mathsf{Hyb}}$ now computes the messages of the protocol π^{RSM} using the simulator algorithms $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3)$ including on each reset execution as done by Sim in the ideal world. $\mathsf{Sim}_{\mathsf{Hyb}}$ also instructs the ideal functionality to deliver outputs to the honest parties as done by Sim. This hybrid

1. **Round 1:** For each honest party P_i, Sim does the following including on each reset:
   - Compute $\mathsf{msg}_{1,i} \leftarrow \mathcal{S}_1(1^\lambda, i)$. For each $j \in [n]$ with $j \neq i$, compute:
     - $\mathsf{c}^j_{1,i} \leftarrow \mathsf{NMCom}^R_1(1^\lambda)$.
     - $(\mathsf{ver}^j_{1,i}, \mathsf{zkst}^j_{1,i}) \leftarrow \mathsf{ZK}_1(1^\lambda)$ and $(\mathsf{ver}^j_{2,i}, \mathsf{zkst}^j_{2,i}) \leftarrow \mathsf{ZK}_1(1^\lambda)$.
   - Send $(\mathsf{msg}_{1,i}, \mathsf{c}^j_{1,i}, \mathsf{ver}^j_{1,i}, \mathsf{ver}^j_{2,i})$ for all $j \in [n]$.
2. **Round 2:** Let τ_1 denote the protocol transcript after round 1. For each honest party P_i, Sim does the following including on each reset:
   - Compute $\mathsf{msg}_{2,i} \leftarrow \mathcal{S}_2(\tau_1, i)$. For each $j \in [n]$ with $j \neq i$, compute:
     - $\mathsf{c}^j_{2,i} \leftarrow \mathsf{NMCom}^S_2(0^{p(\lambda)}, \mathsf{c}^i_{1,j}; \mathsf{r}^j_{\mathsf{c},i})$ using a random string $\mathsf{r}^j_{\mathsf{c},i}$.
     - $\mathsf{prove}^j_{2,i} \leftarrow \mathsf{Sim}^{\mathsf{ZK}}(\mathsf{ver}^i_{1,j}, \mathsf{st}^j_{2,i})$ for $\mathsf{st}^j_{2,i} = (\mathsf{c}^i_{1,j}, \mathsf{c}^j_{2,i}, \mathsf{msg}_{1,i}, \mathsf{msg}_{2,i}, \tau_1) \in L$. This step takes time $\mathsf{T}^{\mathsf{Sim}}_{\mathsf{ZK}}$.
   - Send $(\mathsf{msg}_{2,i}, \mathsf{c}^j_{2,i}, \mathsf{prove}^j_{2,i})$ for all $j \in [n]$.
3. **Query to Ideal Functionality:** Sim does the following:
   - For each honest party P_i and for each $j \in [n]$ with $j \neq i$, do:
     - Abort if $\mathsf{ZK}_3(\mathsf{zkst}^j_{1,i}, \mathsf{st}^i_{2,j}) \neq 1$ where τ_1 is the protocol transcript after round 1 such that $\mathsf{st}^i_{2,j} = (\mathsf{c}^j_{1,i}, \mathsf{c}^i_{2,j}, \mathsf{msg}_{1,j}, \mathsf{msg}_{2,j}, \tau_1)$
     - Compute $(\mathsf{x}^i_j, \mathsf{r}^i_j) = \mathsf{Ext.Com}(\mathsf{c}^j_{1,i}, \mathsf{c}^i_{2,j})$. This step takes time $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$.
   - For each malicious party P_j, do:
     - Output "Special Abort" if $\{(\mathsf{x}^i_j, \mathsf{r}^i_j)\}$ is not equal $\forall$ honest P_i. Set $(\mathsf{x}_j, \mathsf{r}_j) = (x^1_j, \mathsf{r}^1_j)$. Output "Special Abort" if $\mathsf{msg}_{1,j} \neq \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_1(\mathsf{x}_j, \mathsf{r}_j)$ and $\mathsf{msg}_{2,j} \neq \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_2(\mathsf{x}_j, \mathsf{r}_j, \tau_1)$.
     - Send the set $\{\mathsf{x}_j\}$ to the ideal functionality and receive output y.[a]
     - Let R denote the set $\{\mathsf{x}_j, \mathsf{r}_j\}$.
4. **Round 3:** Let τ_2 denote the protocol transcript after round 2. For each honest party P_i, Sim does the following including on each reset: Compute and send $\mathsf{msg}_{3,i} \leftarrow \mathcal{S}_3(\mathsf{y}, \mathsf{R}, \tau_2, i)$ together with $\mathsf{prove}^j_{3,i}$ for $j \in [n], j \neq i$ where $\mathsf{prove}^j_{3,i} \leftarrow \mathsf{Sim}^{\mathsf{ZK}}(\mathsf{ver}^i_{2,j}, \mathsf{st}^j_{3,i})$ for the statement $\mathsf{st}^j_{3,i} = (\mathsf{c}^i_{1,j}, \mathsf{c}^j_{2,i}, \mathsf{msg}_{3,i}, \tau_2) \in L_1$. This step takes time $\mathsf{T}^{\mathsf{Sim}}_{\mathsf{ZK}}$.
5. **Special Abort Phase:** Sim outputs "Special Abort" if, for each malicious party P_j, $\mathsf{msg}_{3,j} \neq \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_3(\mathsf{x}_j, \mathsf{r}_j, \tau_2)$.
6. **Output Computation:** Sim does the following:
   - For each honest P_i, for each $j \in [n]$ with $j \neq i$, abort if $\mathsf{ZK}_3(\mathsf{zkst}^j_{2,i}, \mathsf{st}^i_{3,j}) \neq 1$.
   - Else, instruct the ideal functionality to deliver output to the honest parties.

[a] Each time the adversary resets an honest party before round 3, simulator makes a call to the ideal functionality.

Figure 8: Simulation strategy in the 3 round protocol

is now same as the ideal world. Once again, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$.

We now show that every pair of successive hybrids is computationally indistinguishable.

Lemma B.4. *Assuming soundness of the* $\mathsf{SPSS.ZK}$ *argument system, binding of the non-malleable commitment scheme and correctness of the protocol* π^{RSM}*,* Hyb_1 *is computationally indistinguishable from* Hyb_2*.*

Proof. The only difference between the two hybrids is that in Hyb_2, $\mathsf{Sim}_{\mathsf{Hyb}}$ may output "Special Abort" in some reset execution which doesn't happen in Hyb_1. More specifically, in Hyb_2, "Special Abort" occurs if event E described below is true in some reset execution.

Event E: Is true if : For any malicious party P_j

- All the $\mathsf{SPSS.ZK}$ proofs sent by P_j in round 2 and 3 verify correctly.
  (AND)
- Either of the following occur:
  - The set of values $\{(\mathsf{x}^i_j, \mathsf{r}^i_j)\}$ that are committed to using the non-malleable commitment is not same for every i where P_i is honest. (OR)
  - $\mathsf{msg}_{1,j} \neq \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_1(\mathsf{x}_j, \mathsf{r}_j)$ (OR)
  - $\mathsf{msg}_{2,j} \neq \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_2(\mathsf{x}_j, \mathsf{r}_j, \tau_1)$ where τ_1 is the protocol transcript after round 1. (OR)
  - $\mathsf{msg}_{3,j} \neq \pi^{\mathsf{RSM}}.\mathsf{NextMsg}_3(\mathsf{x}_j, \mathsf{r}_j, \tau_2)$ where τ_2 is the protocol transcript after round 2.

That is, in simpler terms, the event E occurs if for any malicious party, it gives valid ZK proofs in round 2 and 3 but its protocol transcript is not consistent with the values it committed to.

Therefore, in order to prove the indistinguishability of the two hybrids, it is enough to prove the lemma below.

Sub-Claim 1. $\Pr[\text{Event } \mathsf{E} \text{ is true in } \mathsf{Hyb}_2] = \mathsf{negl}(\lambda)$.

Proof. We now prove the sub-lemma. Suppose the event E does occur in some reset execution. From the binding property of the commitment scheme and the correctness of the protocol π^{RSM}, observe that if any of the above conditions are true, it means there exists i, j such that the statement $\mathsf{st}^i_{2,j} = (\mathsf{c}^j_{1,i}, \mathsf{c}^i_{2,j}, \mathsf{msg}_{1,j}, \mathsf{msg}_{2,j}, \tau_1) \notin L$, where P_i is honest and P_j is malicious. However, the proof for the statement verified correctly which means that the adversary has produced a valid proof for a false statement. This violates the soundness property of the SPSSZK argument system which is a contradiction. □

□

Lemma B.5. *Assuming the zero knowledge property of the* $\mathsf{SPSS.ZK}$ *argument system,* Hyb_2 *is computationally indistinguishable from* Hyb_3*.*

Proof. The only difference between the two hybrids is that including every reset execution, in Hyb_2, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the proofs in Rounds 2 and 3 honestly, by running the algorithm ZK_2 of the $\mathsf{SPSS.ZK}$ argument system, whereas in Hyb_3, a simulated proof is used. If the adversary $\mathcal{A}$ can distinguish between the two hybrids, we will use $\mathcal{A}$ to design an algorithm $\mathcal{A}_{\mathsf{ZK}}$ that breaks the zero knowledge property of the argument system.

Suppose the adversary can distinguish between the two hybrids with non-negligible probability p. Then, by a simple hybrid argument, there exists hybrids $\mathsf{Hyb}_{2,k}$ and $\mathsf{Hyb}_{2,k+1}$ that the adversary can distinguish with non-negligible probability $p' < p$ such that: the only difference between the two hybrids is in the proof sent by an honest party P_i to a (malicious) party P_j in one of the rounds . Let's say it is the proof in round 2.

$\mathcal{A}_{\mathsf{ZK}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with $\mathcal{A}$ and performs all the steps exactly as in $\mathsf{Hyb}_{2,k}$ except the proof in Round 2 sent by P_i to P_j. It interacts with a challenger $\mathcal{C}$ of the $\mathsf{SPSS.ZK}$ argument system and sends the first round message $\mathsf{ver}^i_{1,j}$ it received from the adversary. $\mathcal{A}_{\mathsf{ZK}}$ receives from $\mathcal{C}$ a proof that is either honestly computed or simulated. $\mathcal{A}_{\mathsf{ZK}}$ sets this received proof as its message $\mathsf{prove}^j_{i,2}$ in Round 2 of its interaction with $\mathcal{A}$. In the first case, this exactly corresponds to $\mathsf{Hyb}_{2,k}$ while the latter exactly corresponds to $\mathsf{Hyb}_{2,k+1}$. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids, $\mathcal{A}_{\mathsf{ZK}}$ can use the same distinguishing guess to distinguish the proofs: i.e, decide whether the proofs received from $\mathcal{C}$ were honest or simulated. Now, notice that $\mathcal{A}_{\mathsf{ZK}}$ runs only in time $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$ (during the input extraction phase), while the $\mathsf{SPSS.ZK}$ system is secure against adversaries running in time T_{ZK}. Since $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}} < \mathsf{T}_{\mathsf{ZK}}$, this is a contradiction and hence proves the lemma.
In particular, this also means the following: $\Pr[\text{Event E is true in } \mathsf{Hyb}_3] = \mathsf{negl}(\lambda)$. $\square$

Lemma B.6. *Assuming the non-malleability property of the non-malleable commitment scheme* NMCom*,* Hyb_3 *is computationally indistinguishable from* Hyb_4*.*

Proof. We will prove this using a series of computationally indistinguishable intermediate hybrids as follows.

- $\mathsf{Hyb}_{3,1}$**:** This is same as Hyb_3 except that including on each reset execution, the simulator $\mathsf{Sim}_{\mathsf{Hyb}}$ does not run the input extraction phase apart from verifying the $\mathsf{SPSS.ZK}$ proofs. Also, $\mathsf{Sim}_{\mathsf{Hyb}}$ does not run the special abort phase. In particular, the $\mathsf{Ext.Com}$ algorithm is not run and there is no "Special Abort". In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in time $\mathsf{T}^{\mathsf{Sim}}_{\mathsf{ZK}}$ which is lesser than $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$.

- $\mathsf{Hyb}_{3,2}$**:** This hybrid is identical to the previous hybrid except that including on each reset execution, in Round 2, $\mathsf{Sim}_{\mathsf{Hyb}}$ now computes all the messages $(\mathsf{c}^j_{2,i})$ as non-malleable commitments of $0^{p(\lambda)}$ as done by Sim in the ideal world. In this hybrid too, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in time $\mathsf{T}^{\mathsf{Sim}}_{\mathsf{ZK}}$.

- $\mathsf{Hyb}_{3,3}$**:** This is same as Hyb_3 except that including on each reset execution, the simulator does run the input extraction phase and the special abort phase. It is easy to see that $\mathsf{Hyb}_{3,3}$ is the same as Hyb_4. In this hybrid, $\mathsf{Sim}_{\mathsf{Hyb}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$ which is greater than $\mathsf{T}^{\mathsf{Sim}}_{\mathsf{ZK}}$.

We now prove the indistinguishability of these intermediate hybrids and this completes the proof of the lemma.

Sub-Claim 2. Hyb_3 *is statistically indistinguishable from* $\mathsf{Hyb}_{3,1}$*.*

Proof. The only difference between the two hybrids is that in Hyb_3, the simulator might output "Special Abort" which doesn't happen in $\mathsf{Hyb}_{3,1}$. As shown in the proof of Lemma B.5, the probability that Event E occurs in Hyb_3 is negligible. This means that the probability that the simulator outputs "Special Abort" in Hyb_3 is negligible and this completes the proof. $\square$

Sub-Claim 3. *Assuming the non-malleability property of the non-malleable commitment scheme* NMCom*,* $\mathsf{Hyb}_{3,1}$ *is computationally indistinguishable from* $\mathsf{Hyb}_{3,2}$*.*

Proof. The only difference between the two hybrids is that in $\mathsf{Hyb}_{3,1}$, for every honest party P_i, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the commitment messages $(\mathsf{c}^j_{2,i})$ as a commitment of $(\mathsf{x}_i, \mathsf{r}_i)$, whereas in $\mathsf{Hyb}_{3,2}$, they are computed as a commitment of $(0^{p(\lambda)})$. If the adversary $\mathcal{A}$ can distinguish between the two hybrids, we will use $\mathcal{A}$ to design an algorithm $\mathcal{A}_{\mathsf{NMC}}$ that breaks the security of the non-malleable commitment scheme NMCom.

$\mathcal{A}_{\mathsf{NMC}}$ acts as the man-in-the-middle adversary interacting with a challenger $\mathcal{C}$. $\mathcal{A}_{\mathsf{NMC}}$ also plays the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with the adversary $\mathcal{A}$. It generates all the messages except the messages $\mathsf{c}^j_{1,i}$ and $\mathsf{c}^j_{2,i}$ exactly as done by $\mathsf{Sim}_{\mathsf{Hyb}}$ in $\mathsf{Hyb}_{3,1}$. Corresponding to each message $\mathsf{c}^j_{1,i}$ that $\mathcal{A}_{\mathsf{NMC}}$ has to send, it receives one first round message from $\mathcal{C}$ (on the right side) corresponding to the scheme NMCom. $\mathcal{A}_{\mathsf{NMC}}$ forwards these messages to the adversary $\mathcal{A}$ as its first round messages $(\mathsf{c}^j_{1,i})$. Similarly, for each pair of messages $(\mathsf{c}^i_{1,j})$ it receives from $\mathcal{A}$ as part of the first round messages of the scheme NMCom, $\mathcal{A}_{\mathsf{NMC}}$ forwards the messages to $\mathcal{C}$ as its first round messages for the commitment (to the left side respectively). Then, for each $\mathsf{c}^j_{2,i}$ that $\mathcal{A}_{\mathsf{NMC}}$ is supposed to send to $\mathcal{A}$, it receives a second round commitment message from the challenger $\mathcal{C}$. In one case, all of these are commitments to the respective $(\mathsf{x}_i, \mathsf{r}_i)$ values while in the second case, they are all commitments to $(0^{p(\lambda)})$. $\mathcal{A}_{\mathsf{NMC}}$ forwards these messages as its commitment messages $\mathsf{c}^j_{2,i}$ to the adversary $\mathcal{A}$. Once again, it forwards each message $\mathsf{c}^i_{2,j}$ it receives from $\mathcal{A}$, as its second round commitment message in its interaction with the challenger $\mathcal{C}$. That is, these are the commitments on the right side generated by the man-in-the-middle.

Now, we can clearly see that in the first case, when $\mathcal{C}$ generates commitments to $(\mathsf{x}_i, \mathsf{r}_i)$, $\mathcal{A}$'s view corresponds to $\mathsf{Hyb}_{3,1}$ while in the latter case, it exactly corresponds to $\mathsf{Hyb}_{3,2}$. However, from the security of the non-malleable commitment scheme, the joint distribution of the value committed to by the adversary $\mathcal{A}_{\mathsf{NMC}}$ (which is the same as $\mathcal{A}$'s commitments) and its view must be indistinguishable in both cases. Therefore, if $\mathcal{A}$ can distinguish between the two hybrids, then $\mathcal{A}_{\mathsf{NMC}}$ can break the non-malleability property of the commitment scheme NMCom. However, $\mathcal{A}_{\mathsf{NMC}}$ only runs in time $\mathsf{T}^{\mathsf{Sim}}_{\mathsf{ZK}} < \mathsf{T}^{\mathsf{Sec}}_{\mathsf{Com}}$ and hence this is a contradiction, thus proving the sub-lemma.

Also, notice that since the joint distribution of the adversary $\mathcal{A}$'s committed values and his view is indistinguishable in both hybrids, this implies that Event E still occurs only with negligible probability in $\mathsf{Hyb}_{3,2}$ as well. $\square$

Sub-Claim 4. $\mathsf{Hyb}_{3,2}$ *is statistically indistinguishable from* $\mathsf{Hyb}_{3,3}$*.*

Proof. The only difference between the two hybrids is that in $\mathsf{Hyb}_{3,3}$, the simulator might output "Special Abort" which doesn't happen in $\mathsf{Hyb}_{3,2}$. As shown in the proof of Sub-Claim 3, the probability that Event E occurs in $\mathsf{Hyb}_{3,2}$ is negligible. This means that the probability that the simulator outputs "Special Abort" in $\mathsf{Hyb}_{3,3}$ is negligible and this completes the proof. $\square$

$\square$

Lemma B.7. *Assuming the security of the protocol* π^{RSM}*,* Hyb_4 *is computationally indistinguishable from* Hyb_5*.*

Proof. The only difference between the two hybrids is that including on each reset execution, in Hyb_4, $\mathsf{Sim}_{\mathsf{Hyb}}$ computes the messages of protocol π^{RSM} correctly using the honest parties' inputs, whereas in Hyb_5, they are computed by running the simulator $\mathcal{S}$ for protocol π^{RSM}. If the adversary $\mathcal{A}$ can distinguish between the two hybrids, we will use $\mathcal{A}$ to design an algorithm $\mathcal{A}_{\mathsf{SM}}$ that can break the security of protocol π^{RSM}.

$\mathcal{A}_{\mathsf{SM}}$ interacts with a challenger $\mathcal{C}$ to break the security of protocol π^{RSM}. Also, $\mathcal{A}_{\mathsf{SM}}$ performs the role of $\mathsf{Sim}_{\mathsf{Hyb}}$ in its interaction with the adversary $\mathcal{A}$. Whatever parties $\mathcal{A}$ wishes to corrupt, $\mathcal{A}_{\mathsf{SM}}$ corrupts the same parties in its interaction with π^{RSM}. Similarly, whatever messages $\mathcal{A}$ sends to $\mathcal{A}_{\mathsf{SM}}$ as part of the protocol π that correspond to π^{RSM} messages, $\mathcal{A}_{\mathsf{SM}}$ sends the same messages to the challenger $\mathcal{C}$. Now, whatever messages $\mathcal{C}$ sends, $\mathcal{A}_{\mathsf{SM}}$ forwards the same to the adversary $\mathcal{A}$ as its messages for the π^{RSM} protocol. $\mathcal{A}_{\mathsf{SM}}$ does everything else exactly as in Hyb_5. Whenever $\mathcal{A}$ wants to reset any party, $\mathcal{A}_{\mathsf{SM}}$ resets that party in its interaction with $\mathcal{C}$.

Observe that $\mathcal{A}_{\mathsf{SM}}$ runs in time $\mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$. If $\mathcal{C}$ sends messages that are computed correctly, this exactly corresponds to Hyb_4 in $\mathcal{A}_{\mathsf{SM}}$'s interaction with $\mathcal{A}$. On the other hand, if $\mathcal{C}$ sends simulated messages, this exactly corresponds to Hyb_5. Therefore, if $\mathcal{A}$ can distinguish between these two hybrids, $\mathcal{A}_{\mathsf{SM}}$ can use the same distinguishing guess to break the security of protocol π^{RSM}. However, π^{RSM} is secure against all adversaries running in time T_{SM}, where $\mathsf{T}_{\mathsf{SM}} > \mathsf{T}^{\mathsf{Brk}}_{\mathsf{Com}}$ and hence this is a contradiction. This completes the proof of the lemma. $\square$