On the Power of Secure Two-Party Computation

Carmit Hazay* Muthuramakrishnan Venkitasubramaniam†

Abstract

Ishai, Kushilevitz, Ostrovsky and Sahai (STOC'07, SIAM JoC 2009) introduced the powerful "MPC-in-the-head" technique that provided a general transformation of information-theoretic MPC protocols secure against passive adversaries to a ZK proof in a "black-box" way. In this work, we extend this technique and provide a generic transformation of any semi-honest secure two-party computation (2PC) protocol (with mild adaptive security guarantees) in the so called *oblivious-transfer* hybrid to an adaptive ZK proof for any NP-language, in a "black-box" way assuming only one-way functions. Our basic construction based on Goldreich-Micali-Wigderson's 2PC protocol yields an adaptive ZK proof with communication complexity proportional to quadratic in the size of the circuit implementing the NP relation. Previously such proofs relied on an expensive Karp reduction of the NP language to Graph Hamiltonicity (Lindell and Zarosim (TCC'09, Journal of Cryptology 2011)). We also improve our basic construction to obtain the first linear-rate adaptive ZK proofs by relying on efficient maliciously secure 2PC protocols. Core to this construction is a new way of transforming 2PC protocols to efficient (adaptively secure) instance-dependent commitment schemes.

As our second contribution, we provide a general transformation to construct a randomized encoding of a function f from any 2PC protocol that securely computes a related functionality (in a black-box way). We show that if the 2PC protocol has mild adaptive security guarantees then the resulting randomized encoding (RE) can be decomposed to an offline/online encoding.

As an application of our techniques, we show how to improve the construction of Lapidot and Shamir (Crypto'90) to obtain "input-delayed" ZK proofs which are proofs where the honest prover's algorithm does not require the actual statement until the last round. Our transformation also yields the simplest constructions of both static and adaptive ZK proofs from standard 2PC protocols of Yao and Goldreich-Micali-Wigderson.

Keywords: adaptive zero-knowledge proofs, secure two-party computation, randomized encoding, interactive hashing, instance-dependent commitments

*Bar-Ilan University, Israel. Email: carmit.hazay@biu.ac.il.

†University of Rochester, Rochester, NY 14611, NY. Email: muthuv@cs.rochester.edu.

Contents

1 Introduction

In this work we establish new general connections between three fundamental tasks in cryptography: secure two-party computation, zero-knowledge proofs and randomized encoding. We begin with some relevant background regarding each of these tasks.

Secure multiparty computation. The problem of *secure multiparty computation* (MPC) [Yao82, CCD87, GMW87, BGW88] considers a set of parties with private inputs that wish to jointly compute some function of their inputs while preserving certain security properties. Two of these properties are *privacy*, meaning that the output is learned but nothing else, and *correctness*, meaning that no corrupted party or parties can cause the output to deviate from the specified function. Security is formalized using the simulation paradigm where for every adversary $\mathcal{A}$ attacking a real protocol, we require the existence of a simulator $\mathcal{S}$ that can cause the same damage in an ideal world, where an incorruptible trusted third party computes the function for the parties and provides them their output.

Honest vs. dishonest majority. Generally speaking, there are two distinct categories for MPC protocols: (1) one for which security is guaranteed only when a *majority* of the parties are honest, and (2) one for which security is guaranteed against an arbitrary number of corrupted parties. In the former category it is possible to construct "information-theoretic" secure protocols where security holds unconditionally,[1] whereas in the latter only computational security can be achieved while relying on cryptographic assumptions.[2] The former setting necessarily requires 3 or more parties while the latter can be constructed with just two parties. In this work, we will focus on the latter setting, considering secure two-party computation.

Semi-honest vs malicious adversary. The adversary may be *semi-honest*, meaning that it follows the protocol specification but tries to learn more than allowed, or *malicious*, namely, arbitrarily deviating from the protocol specification in order to compromise the security of the other players in the protocol. Constructing semi-honestly secure protocols is a much easier task than achieving security against a malicious adversary.

Static vs. adaptive corruption. The initial model considered for secure computation was one of a *static adversary* where the adversary controls a subset of the parties (who are called *corrupted*) before the protocol begins, and this subset cannot change. A stronger corruption model allows the adversary to choose which parties to corrupt throughout the protocol execution, and as a function of its view; such an adversary is called *adaptive*. Adaptive corruptions model "hacking" attacks where an external attacker breaks into parties' machines in the midst of a protocol execution and are much harder to protect against. In particular, protocols that achieve adaptivity are more complex and the computational hardness assumptions needed seem stronger; see [CLOS02, KO04, CDD$^+$04, IPS08]. Achieving efficiency seems also to be much harder.

Zero-knowledge. Zero-knowledge (ZK) interactive protocols [GMR89] are paradoxical constructs that allow one party (denoted the prover) to convince another party (denoted the verifier) of the validity of a mathematical statement $x \in \mathcal{L}$, while providing *zero additional knowledge* to the verifier. Beyond being fascinating in their own right, ZK proofs have numerous cryptographic applications and are one of the most fundamental cryptographic building blocks. The zero-knowledge property is formalized using the *simulation paradigm*. That is, for every malicious verifier $\mathcal{V}^*$, we require the existence of a simulator $\mathcal{S}$ that reproduces a view of $\mathcal{V}^*$ that is indistinguishable from a view when interacting with the honest prover, given only the

[1] Namely, against computationally unbounded adversaries.

[2] If one is willing to provide ideal access to an oblivious-transfer functionality then one can achieve information-theoretic security even in the honest minority setting [GMW87, CvdGT95, IPS08].

input x. Zero-knowledge protocols can be viewed as an instance of secure two-party computation where the function computed by the third-party simply verifies the validity of a witness held by the prover.

Static vs. adaptive. Just as with secure computation, the adversary in a zero-knowledge protocol can be either static or adaptive. Security in the presence of a statically corrupted prover implies that the protocol is sound, namely, a corrupted prover cannot convince a verifier of a false statement. Whereas security in the presence of a statically corrupted verifier implies that the protocol preserves zero-knowledge. Adaptive security on the other hand requires a simulator that can simulate the corruptions of both the prover and verifier.

Much progress has been made in constructing highly efficient ZK proofs in the static setting. In a recent breakthrough result, Ishai, Kushilevitz, Ostrovsky and Sahai [IKOS09] provided general constructions of ZK proofs for any NP relation $\mathcal{R}(x, \omega)$ which make a "black-box" use of an MPC protocol for a related multiparty functionality f, where by black-box we mean that f can be programmed to make only black-box (oracle) access to the relation $\mathcal{R}$. Leveraging the highly efficient MPC protocols in the literature [DI06] they obtained the first "constant-rate" ZK proof. More precisely, assuming one-way functions, they showed how to design a ZK proof for an arbitrary circuit C of size s and bounded fan-in, with communication complexity $O(s) + \mathsf{poly}(\kappa, \log s)$ where κ is the security parameter. Besides this, the work of [IKOS09] introduced the very powerful "MPC-in-the-head" technique that has found numerous applications in obtaining "black-box" approaches, such as unconditional two-party computation [IPS08], secure computation of arithmetic circuits [IPS09], non-malleable commitments [GLOV12], zero-knowledge PCPs [IW14], resettably-sound ZK [OSV15] to name a few, as well as efficient protocols, such as OT-based cryptography [HIKN08, IPS08, IPS09] and homomorphic UC commitments [CDD+15].

In contrast, in the adaptive setting, constructing adaptive zero-knowledge proofs is significantly harder and considerably inefficient. Beaver [Bea96] showed that unless the polynomial hierarchy collapses the ZK proof of [GMR89] is not secure in the presence of adaptive adversaries. Quite remarkably, Lindell and Zarosim showed in [LZ11] that adaptive zero-knowledge proofs for any NP language can be constructed assuming only one-way functions. However, it is based on reducing the statement that needs to be proved to an NP complete problem, and is rather inefficient. In fact, the communication complexity of the resulting zero knowledge is $O(s^4)$ where s is the size of the circuit. Moreover, the resulting protocol will not be "black-box" in the NP relation as in the constructions of [IKOS09]. A first motivation for our work is the goal of finding alternative "black-box" approaches of constructing (efficient) adaptive ZK proofs.

Randomized Encoding (RE). The third fundamental primitive considered in this work is *randomized encoding* (RE). Formalized in the works of [IK00, IK02, AIK06], randomized encoding explores to what extent the task of securely computing a function can be simplified by settling for computing an "encoding" of the output. Loosely speaking, a function $\widehat{f}(x, r)$ is said to be a randomized encoding of a function f if the output distribution depends only on $f(x)$. More formally, the two properties required of a randomized encoding are: (1) given the output of $\widehat{f}$ on (x, r), one can efficiently compute (decode) $f(x)$, and (2) given the value $f(x)$ one can efficiently sample from the distribution induced by $\widehat{f}(x, r)$ where r is uniformly sampled. One of the earliest constructions of a randomized encoding is that of "garbled circuits" and originates in the work of Yao [Yao86]. Additional variants have been considered in the literature in the early works of [Kil88, FKN94]. Since its introduction, randomized encoding has found numerous applications, especially in parallel cryptography where encodings with small parallel complexity yields highly efficient secure computation [IK00, IK02, AIK06]. (See also [GKR08, GGP10, AIK10, GIS+10, BHHI10, BHR12, App14] for other applications)

Online/offline complexity. In an online/offline setting [AIKW13], one considers an encoding $\widehat{f}(x, r)$

which can be split as an offline part $\widehat{f}_{\text{OFF}}(r)$ which only depends on the function f, and an online part $\widehat{f}_{\text{ON}}(x,r)$ that additionally depends on input x. This notion is useful in a scenario where a weak device is required to perform some costly operation f on sensitive information x: In an offline phase $\widehat{f}_{\text{OFF}}(r)$ is published or transmitted to a cloud, and later in an online phase, the weak device upon observing the sample x, transmits the encoding $\widehat{f}_{\text{ON}}(x,r)$. The cloud then uses the offline and online parts to decode the value $f(x)$ and nothing else. The goal in such a setting is to minimize the online complexity, namely the number of bits in $\widehat{f}_{\text{ON}}(x,r)$. In the classic garbled circuit construction, the online complexity is proportional to $|x|\mathsf{poly}(\kappa)$ where κ is the security parameter. More recently, Applebaum, Ishai, Kushilevitz and Waters showed in [AIKW13] how to achieve constant online rate of $(1+o(1))|x|$ based on concrete number-theoretic assumptions.

A notoriously hard question here is to construct an *adaptively secure* RE where privacy is maintained even if the online input x is *adaptively* chosen based on the offline part. In fact, the standard constructions of garbled circuits (with short keys) do not satisfy this stronger property unless some form of "exponentially-hard" assumption is made [GKR08] or analyzed in the presence of the so-called *programmable random-oracle* model [AIKW13]. In fact, recently, it was shown in [AIKW13] that any adaptively secure randomized encoding must have an online complexity proportional to the output length of the function.

While the connection between RE and secure computation has been explored only in one direction, where efficient RE yield efficient secure computation, we are not aware of any implication in the reverse direction. A second motivation of our work is to understand this direction while better understanding the complexity of constructing secure protocols by relying on the lower bounds already established for the simpler RE primitive.

1.1 Our Contribution

In this work we present the following transformations:

1. A general construction of a *static* zero-knowledge proof system $\Pi_{\mathcal{R}}$ for any NP relation $\mathcal{R}(x,\omega)$ that makes a black-box use[3] of a two-party protocol Π_f^{OT}, carried out between parties P_1 and P_2, for a related functionality f in the oblivious-transfer (OT) hybrid model,[4] along with a (statically secure) bit commitment protocol,[5] that can be realized assuming only one-way functions. The requirements on our protocol Π_f^{OT} are: Perfect (UC) security against static corruptions by semi-honest adversaries. For example, the standard versions of the known [GMW87] protocol (denoted by GMW) and [Yao86]'s protocol satisfy these requirements. We remark here that the approach of [IKOS09] that transforms general MPC protocol is inapplicable here, as their work crucially requires the MPC protocol to admit at least three or more parties.

2. A general construction of an *adaptively secure* zero-knowledge proof system $\Pi_{\mathcal{R}}$ for any NP relation $\mathcal{R}(x,\omega)$ that makes a black-box use of a two-party protocol Π_f^{OT}, carried out between parties P_1 and P_2, for a related functionality f in the oblivious-transfer (OT) hybrid model, along with a (statically secure) bit commitment protocol, that can be realized assuming only one-way functions. The requirements on our protocol Π_f^{OT} are: (1) Perfect (UC) security against semi-honest parties admitting a

[3] The functionality f can be efficiently defined by making only a black-box (oracle) access to the NP relation $\mathcal{R}$.

[4] Where all parties have access to an idealized primitive that implements the OT functionality, namely, the functionality upon receiving input (s_0, s_1) from the sender and a bit b from the receiver, returns s_b to the receiver and nothing the sender.

[5] We will be able to instantiate our commitment schemes using a statistically-binding commitment scheme for commitments made by the prover in the ZK protocol, and by a statistically-hiding commitment scheme for commitments made by the verifier.

static corruption of P_1 and an adaptive corruption of P_2, and (2) P_1 is the sender in all OT invocations. We remark that the semi-honest version of the GMW protocol satisfies these requirements. In fact, we will only require milder properties than perfect privacy (namely, robustness and invertible sampleability) and adaptive corruption (namely, semi-adaptive) which will be further satisfied by the standard Yao's protocol [Yao86] based on garbled circuits.

3. A general construction of a *randomized encoding* for any function f that makes a black-box use (a la [IKOS09]) of a two-party computation protocol Π_f^{OT}, carried out between parties P_1 and P_2, for a related functionality g in the OT-hybrid assuming only one-way functions. If we start with the same requirements as our first transformation (namely, only security against static adversaries) then we obtain a standard randomized encoding. However, if we start with a protocol as required in our second transformation with the additional requirement that it admits (full) adaptive corruption of P_2, we obtain an online/offline RE. Moreover, our construction makes a black-box use of a randomized encoding for the functionality f. Finally, we also show how to obtain an adaptive ZK proof for an NP relation $\mathcal{R}$ using a slightly stronger version of RE (that our second instantiation above will satisfy). An important corollary we obtain here is that starting from an RE that is additionally secure against adaptive chosen inputs we obtain the—so called—input-delayed ZK proof in the static setting.

Next, we provide an overview of our transformations.

Static ZK via (semi-honest) 2PC or "2PC-in-the-head". We begin with a perfectly-correct 2PC protocol Π_f between parties P_1 and P_2 that securely implements the following functionality f: $f(x, \omega_1, \omega_2)$ outputs 1 if and only if $(x, \omega_1 \oplus \omega_2) \in \mathcal{R}$ where ω_1 and ω_2 are the private inputs of P_1 and P_2 in the two party protocol Π_f. We require that the 2PC protocol admits semi-honest UC security against static corruption of P_1 and P_2. Our first step in constructing a ZK proof involves the prover P simulating an honest execution between P_1 and P_2 by first sampling ω_1 and ω_2 at random such that $\omega_1 \oplus \omega_2 = \omega$, where ω is the witness to the statement x and then submitting the transcript of the interaction to the verifier V. The verifier responds with a bit b chosen at random. The prover then reveals the view of P_1 if $b = 0$ and the view of P_2 if $b = 1$, namely it just provides the input and randomness of the respective parties. Soundness follows from the perfect correctness of the protocol. Zero-knowledge, on the other hand, is achieved by invoking the simulation of parties P_1 and P_2 depending on the guess that the simulator makes for the verifier's bit b.

This general construction, however, will inherit the hardness assumptions required for the 2PC, which in the case of [Yao86] and [GMW87] protocols will require the existence of an oblivious-transfer protocol. We next show how to modify the construction to rely only on one-way functions. The high-level idea is that we encode the transcript of all oblivious-transfer invocations by using a randomized encoding of the oblivious-transfer functionality based on one-way functions as follows:

- For every OT call where P_1's input is (s_0, s_1) and P_2's input is t, we incorporate it in the transcript τ by generating a transcript containing the commitments c_0 and c_1 of s_0 and s_1 using a statistically binding commitment scheme com (which can be based on one-way functions), placing the decommitment information of c_t in P_2's random tape.[6]

This protocol results in an interactive commitment phase as we rely on a statistically-binding commitment scheme and the first message corresponding to all commitments needs to be provided by the receiver.

[6]Note that, in Naor's statistically binding commitment scheme [Nao91] the decommitment information is the inverse under a pseudorandom generator that is uniformly sampled, and hence can be placed in the random tape.

Adaptive ZK via "2PC-in-the-head". First, we recall the work of Lindell and Zarosim [LZ11] that showed that constructing adaptively secure ZK proofs can be reduced to constructing *adaptive instance-dependent commitment* schemes [BMO90, IOS97, OV08, LZ11]. In fact, by simply instantiating the commitments from the prover in the (static) ZK proofs of [IKOS09] with instance-dependent commitments, we can obtain an adaptive ZK proof. Briefly, instance-dependent commitment schemes are defined with respect to a language $\mathcal{L} \in \mathsf{NP}$ such that for any statement x the following holds. If $x \in \mathcal{L}$ then the commitment associated with x is computationally hiding, whereas if $x \notin \mathcal{L}$ then the commitment associated with x is perfectly binding. An adaptively secure instance-dependent commitment scheme additionally requires that there be a "fake" commitment algorithm which can be produced using only the statement x, but later, given a witness ω such that $(x, \omega) \in \mathcal{R}$, be opened to both 0 and 1.

First, we describe an instance-dependent commitment scheme using a (perfectly-correct) 2PC protocol Π_f engaged between parties P_1 and P_2 that securely implements the following functionality f: $f(x, \omega_1, \omega_2)$ outputs 1 if and only if $(x, \omega_1 \oplus \omega_2) \in \mathcal{R}$ where ω_1 and ω_2 are the private inputs of P_1 and P_2 in the two party protocol Π_f. We will require that only P_2 receives an output and that Π_f is (UC) secure against the following adversaries: (1) A semi-honest adversary $\mathcal{A}_1$ that statically corrupts P_1, and (2) A semi-honest adversary $\mathcal{A}_2$ that statically corrupts P_2.

Given such a 2PC Π_f a commitment to the message 0 is obtained by committing to the view of party P_1 in an interaction using Π_f, using the simulator $\mathcal{S}_1$ for adversary $\mathcal{A}_1$ as follows. The commitment algorithm runs $\mathcal{S}_1$ on input a random string ω_1 that serves as the input of P_1. The output of the commitment on input 0 is τ where τ is the transcript of the interaction between P_1 and P_2 obtained from the view of P_1 generated by $\mathcal{S}_1$. A commitment to 1 is obtained by running the simulator $\mathcal{S}_2$ corresponding to $\mathcal{A}_2$ where the input of P_2 is set to a random string ω_2. The output of the commitment is transcript τ obtained from the view of P_2 output by $\mathcal{S}_2$. Decommitting to 0 simply requires producing input and output (ω_1, r_1) for P_1 such that the actions of P_1 on input ω_1 and random tape r_1 are consistent with the transcript τ. Decommitting to 1 requires producing input and randomness (ω_2, r_2) for P_2 consistent with τ and P_2 outputs 1 as the output of the computation. The hiding property of the commitment scheme follows from the fact that the transcript does not reveal any information regarding the computation. The binding property for statements $x \notin \mathcal{L}$, on the other hand, relies on the perfect correctness of the protocol. More precisely, if a commitment phase τ is decommitted to both 0 and 1, then we can extract inputs and randomness for P_1 and P_2 such that the resulting interaction with honest behavior yields τ as the transcript of messages exchanged and P_2 outputting 1. Note that this is impossible since the protocol is perfectly correct and 1 is not in the image of f for $x \notin \mathcal{L}$.

Next, to obtain an *adaptively secure* instance-dependent commitment scheme we will additionally require that Π_f be secure against a semi-honest adversary $\mathcal{A}_3$ that first statically corrupts P_1 and then adaptively corrupts P_2 at the end of the execution. This adversary is referred to as a semi-adaptive adversary in the terminology of [GWZ09]. The fake commitment algorithm follows the same strategy as committing to 0 with the exception that it relies on the simulator $\mathcal{S}_3$ of $\mathcal{A}_3$. $\mathcal{S}_3$ is a simulator that first produces a view for P_1 and then post execution produces a view for P_2. More formally, the fake commitment algorithm sets P_1's input to a random string ω_1 and produces P_1's view using $\mathcal{S}_3$ and outputs τ where, τ is the transcript of the interaction. Decommitting to 0 follows using the same strategy as the honest decommitment. Decommitting to 1, on the other hand, requires producing input and randomness for P_2. This can achieve by continuing the simulation by $\mathcal{S}_3$ post execution. However, to run $\mathcal{S}_3$ it needs to produce an input for party P_2 such that it outputs 1. This is possible as the decommitting algorithm additionally receives the real witness ω for x, using which it sets P_2's input as $\omega_2 = \omega \oplus \omega_1$.

In fact, we will only require adversaries $\mathcal{A}_2$ and $\mathcal{A}_3$, as the honest commitment to 0 can rely on $\mathcal{S}_3$.

Indistinguishability of the simulation will then follow by comparing the simulations by $\mathcal{S}_2$ and $\mathcal{S}_3$ with a real-world experiment with adversaries $\mathcal{A}_2, \mathcal{A}_3$ where the parties inputs are chosen at random subject to the condition that they add up to ω and using the fact that the adversaries are semi-honest.

We will follow an approach similar to our previous transformation to address calls to the OT functionality. We will additionally require that P_1 plays the sender's role in all OT invocations. We note that our encoding accommodates an adaptive corruption of P_2, as it enables us to equivocate the random tape of P_2 depending on its input t.

To instantiate our scheme, we can rely on [Yao86] or [GMW87] to obtain an adaptive instance-dependent commitment scheme. Both commitments results in a communication complexity of $O(s\mathsf{poly}(\kappa))$ where s is the size of the circuit implementing the relation $\mathcal{R}$ and κ is the security parameter. Achieving adaptive zero-knowledge is then carried out by plugging in our commitment scheme into the prover's commitments in the [IKOS09] zero-knowledge (ZK) construction, where it commits to the views of the underlying MPC protocol. The resulting protocol will have a complexity of $O(s^2\mathsf{poly}(\kappa))$ and a negligible soundness error. We remark that this construction already improves the previous construction of Lindell and Zarosim that requires the expensive Karp reduction to Graph Hamiltonicity. Our main technical contribution is showing how we can further improve our basic construction to achieve a complexity of $O(s\mathsf{poly}(\kappa))$ and therefore obtaining a "linear"-rate *adaptive* ZK proof.

RE from (semi-honest) 2PC. To construct a RE for a function f, we consider an arbitrary 2PC protocol that securely realizes the related function g that is specified as follows: $g(a_1, a_2) = f(a_1 \oplus a_2)$ where a_1 and a_2 are the private inputs of P_1 and P_2 in the two party protocol Π_g. We will make the same requirements on our 2PC as in the previous case, namely, security with respect to adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$. The offline part of our encoding function $\widehat{f}_{\text{OFF}}(r)$ is defined using the simulator $\mathcal{S}_3$ for adversary $\mathcal{A}_3$ that proceeds as follows. Upon corrupting P_1, $\mathcal{S}_3$ is provided with a random input string a_1, where the simulation is carried out till the end of the execution and temporarily stalled. The output of $\widehat{f}_{\text{OFF}}(r)$ is defined to be the simulated transcript of the interaction between parties P_1 and P_2. Next, upon receiving the input x, the online part $\widehat{f}_{\text{ON}}(x, r)$ continues the simulation by $\mathcal{S}_1$ which corrupts P_2 post execution (at the end of the protocol execution), where P_2's input is set as $a_2 = x \oplus a_1$ and its output is set as $f(x)$. Finally, the output of $\widehat{f}_{\text{ON}}(x, r)$ is defined by the input and random tape of P_2. In essence, $\widehat{f}(x, r) = (\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r))$ constitutes the complete view of P_2 in an execution using Π_g. The decoder simply follows P_2's computation in the view and outputs P_2's output, which should be $f(x)$ by the correctness of the algorithm. The simulation for our randomized encoding $\mathcal{S}$ relies on the simulator for the adversary $\mathcal{A}_2$, denoted by $\mathcal{S}_2$. Namely, upon receiving $f(x)$, $\mathcal{S}$ simply executes $\mathcal{S}_2$. Recalling that $\mathcal{S}_2$ corrupts P_2, $\mathcal{S}$ simply provides a random string a_2 as its input and $f(x)$ as the output. Finally, the offline and online parts are simply extracted from P_2's view accordingly. Privacy will follow analogously as in our previous case.

Note that the offline complexity of our construction is equal to the communication complexity of the underlying 2PC protocol Π_g, whereas the online complexity amounts to the input plus the randomness complexity of P_2. The efficiency of our randomized encoding ties the offline part with the static simulation of party P_1 and the online part with the semi-adative simulation of P_2. Moreover, this protocol can be instantiated by the [Yao86] and [GMW87] protocols, where the OT sub-protocols are implemented using one-way functions as specified before. We remark that the protocol of [Yao86] does not, in general, admit adaptive corruptions, yet it is secure in the presence of a semi-adaptive adversary that adaptively corrupts P_2 after statically corrupting P_1. The [Yao86] based protocol will result in an offline complexity of $O(s\mathsf{poly}(\kappa))$ and an online complexity of $O(n\mathsf{poly}(\kappa))$ where s is the size of the circuit implementing f and n is the input

length.[7] Whereas the [GMW87] protocol will result in an offline and online complexities of $O(s\mathsf{poly}(\kappa))$. While this might not be useful in the "delegation of computation" application of randomized encoding as the online encoding is not efficient, it can be used to construct an instance-dependent commitment scheme where we are interested only in the total complexity of the encoding. Finally, we remark that if we are not interested in an offline/online setting and just require a standard randomized encoding we will requite Π_f to be secure only against a static corruption of P_2 by $\mathcal{A}_2$ and the honest encoding can be carried out by emulating the real world experiment (as opposed to relying on the simulation by $\mathcal{S}_3$).

Next, we provide a construction of instance-dependent commitments based on online/offline RE. Standard RE will not be sufficient for this and we introduce a stronger notion of *robustness* for RE and show that the preceeding construction already satisfies this. Then based on a robust RE we show how to get an instant-dependent commitment scheme. In fact, we can get an adaptive instance-dependent commitment scheme if the underlying RE has a corresponding adaptive property. Since adaptive instance-dependent comitment schemes are sufficient to realize adaptive ZK, this provides a transformation from RE to adaptive ZK.

"Linear"-rate adaptive ZK proof from malicious 2PC. The main drawback in our first construction of adaptive ZK proofs was in the equivocation parameter of our instance-dependent commitment. Namely, to equivocate one bit, we incurred a communication complexity of $O(s\mathsf{poly}(\kappa))$. To improve the communication complexity one needs to directly construct an instance-dependent commitment scheme for a larger message space $\{0,1\}^\ell$. We show how to construct a scheme where the communication complexity depends only *additively* on the equivocation parameter, implying $O((s+\ell)\mathsf{poly}(\kappa))$ overhead. Combining such a scheme with the [IKOS09] ZK proof results in a protocol with communication complexity of $O(ns\mathsf{poly}(\kappa) + \sum_{i=1}^{n} \ell_i \mathsf{poly}(\kappa))$ where ℓ_i is the length of the i^{th} commitment made by the prover. Setting $n = \omega(\log k)$ results in an adaptive ZK proof with negligible soundness error and complexity $O(s\mathsf{poly}(\kappa))$.

Our approach to construct an instance-dependent commitment scheme for larger message spaces is to rely on a maliciously secure two-party computation. Specifically, suppose that for a polynomial-time computable Boolean function $f(x,y)$ we have a 2PC protocol Π_f with parties P_1 and P_2, where P_2 receives the output of the computation and satisfies all the conditions required in our original transformation. In addition we require it to satisfy statistical security against a malicious P_1 (in the OT-hybrid). In fact, it suffices for the protocol to satisfy the following "soundness" condition: If there exists no pair of inputs x, y such that $f(x,y) = 1$ then for any malicious P_1^*, the probability that an honest P_2 outputs 1 is at most 2^{-t}, where the probability is taken over the randomness of party P_2. Then, using such a protocol, we can provide a framework to construct an instance-dependent commitment scheme where the soundness translates to the equivocation parameter, namely, it will be $O(t)$ for soundness 2^{-t}.

Concretely, given an input statement x we consider a protocol Π_f that realizes function f defined as follows: $f(\omega_1, \omega_2) = 1$ iff $(x, \omega_1 \oplus \omega_2) \in \mathcal{R}$. We first describe an (incorrect) algorithm as a stepping stone towards explaining the details of the final construction. The commitment algorithm on input a message m, (just as in our transformation to RE) invokes the simulator $\mathcal{S}_2$ that corresponds to the adversary $\mathcal{A}_2$, which statically corrupts P_2 with an input set to a random string ω_2 and output 1. Upon completing the simulation, the committer submits to the receiver the transcript of the interaction and $\mathsf{Ext}(r_2) \oplus m$ where r_2 is the randomness of P_2 output by the simulation and $\mathsf{Ext}(\cdot)$ is a randomness extractor that extracts $R - O(t)$ bits where R is the length of P_2's random tape. A decommitment simply produces m along with P_2's input and randomness corresponding to the transcript output in the commitment phase. Intuitively, binding follows directly from the soundness condition as no adversarial committer can produce two different random strings for P_2, as Ext extracts out all "accessible" random tapes for P_2. The fake commitment, on the other hand,

[7] We note that the online complexity can be improved by relying on the work of [AIKW13].

relies as above on a simulator corresponding to adversary $\mathcal{A}_1$ that statically corrupts P_1 and adaptively corrupts P_2, where instead of $\mathsf{Ext}(r_2) \oplus m$ it simply sends a random string. Equivocation, on the other hand, is achievable if the simulation can additionally access the entire space of consistent random tapes of P_2 and invert Ext. Several problems arise when materializing this basic framework.

The first issue is that we cannot rely on an extractor as the adversary can adaptively decide on r_2 given the description of Ext. Now, since extractors are only statistically secure, this implies that for certain (very small) set of values for r_2 there could be multiple pre-images with respect to Ext. Instead, we rely on an *interactive hashing* protocol [NOVY98, DHRS04, HR07] that guarantees binding against computationally unbounded adversaries. More precisely, an interactive hashing protocol ensures that if the set of random tapes accessible to the adversary is at most $2^{R-O(t)}$ then except with negligible probability it cannot obtain two random tapes that are consistent with the transcript of the hashing protocol. This protocol will additionally require to satisfy an invertible sampleability property where given an interaction it is possible to compute efficiently a random input consistent with the transcript. We will not be able to rely on the efficient 4-message protocol of [DHRS04] but will rely on the original protocol of [NOVY98] that proceeds in a linear number of rounds (linear in the message length) where inverting simply requires solving a system of linear equations in a finite field.

Another major issue is that the space of consistent random tapes might not be "nice" to be invertible. More precisely, to adaptively decommit a fake commitment to an arbitrary message we require that the space of consistent random tapes for P_2, i.e. consistent with both the transcript τ of the protocol and the transcript of the interactive-hashing protocol in the commitment phase, to be "uniform". We thus consider a variant of the protocol in [IPS08] so that the space of consistent random tapes will be uniform over the bits of a specified length. While this modification solves the problem of "nice" random tapes, it requires re-establishing a certain "soundness" condition in the compilation of [IPS08].

As mentioned before we combine our adaptive instance-dependent commitment scheme with the ZK protocol of [IKOS09]. We will rely on a variant where the MPC protocol in their construction will be instantiated with the classic [BGW88] protocol, as opposed to highly-efficient protocol of [DI06]. The reason is that we will additionally require a reconstructability property[8] of the MPC protocol that can be shown to be satisfied by [BGW88]. Secondly, relying on this efficient variant anyway does not improve the asymptotic complexity to beyond a linear-rate. As an independent contribution we also provide a simple adaptive ZK protocol based on garbled circuits that satisfies reconstructability but will only achieve soundness error $1/2$ (see Section 7.1).

1.2 Applications

We list a few of the applications of our techniques and leave it as future work to explore the other ramifications of our transformations.

Arithmetic circuits and other efficient garbling: As our transformation works for an arbitrary 2PC protocol with the respective properties, we can obtain efficient adaptive ZK proofs for arithmetic circuits by relying on the work of [IPS09]. We explore the concrete parameters in the full version of the paper. Moreover, recently there has been an explosion of works trying to improve the efficiency of garbling schemes that can be used to instantiate very efficient adaptive ZK according to our constructions.

Input-delayed ZK proofs. In [LS90], Lapidot and Shamir provided a three-round witness-indistinguishable (WI) proof of knowledge for Graph Hamiltonicity with a special "input-delayed" property: namely, the

[8] Informally, reconstructability requires that given the views of t out of n players in an instance of the protocol, and the inputs of all parties, it is possible to reconstruct the views of the remaining parties consistent with views of the t parties.

prover uses the statement to be proved only in the last round. Recently, in [CPS+15] it was shown how to obtain efficient input-delayed variants of the related "Sigma protocols" when used in a restricted setting of an OR-composition. We show that starting from a robust RE that is additionally secure against adaptive inputs, we can obtain constructions of input-delayed zero-knowledge proofs that yield an efficient version of the protocol of [LS90]. In essence, the communication complexity linearly depends on the size of the circuit implementing the NP relation. As in our other transformation, this transformation will only depend on the relation in a black-box way. Finally, we show how to realize robust RE secure against adaptive inputs based on recent work of Hemenway et al. [HJO+15].

Instance-dependent trapdoor commitment schemes. As a side result, we show that our constructions imply instance-dependent trapdoor commitment schemes, for which the witness ω serves as a trapdoor that allows to equivocate the commitment into any value from the message space. More specifically, this notion implies the same hiding/binding properties as any instance-dependent commitment scheme with the additional property that the witness allows to decommit a commitment into any message. To the best of our knowledge, our construction is the first trapdoor commitment for all NP. Prior constructions were known only for Σ-protocols [Dam10].

1.3 Perspective

Our work is similar in spirit to the work of [IKOS09, IPS08] where they demonstrate the power information-theoretic MPC protocols in constructing statically-secure protocols. Here, we show the power of (adaptively-secure) 2PC protocols in the OT-hybrid helps in constructing adaptively-secure protocols and randomized encodings. Instantiating our 2PC with the standard protocols of [Yao86] and [GMW87] yields simple constructions of adaptive ZK proofs and randomized encodings. While ZK can be viewed as a special instance of a two-party computation protocol, the resulting instantiation requires stronger assumptions (such as enhanced trapdoor permutations). On the other hand, our transformation requires only one-way functions. As mentioned earlier, we not only provide adaptive ZK proofs, but we obtain two new simple static ZK proofs from our instance-based commitments.

A second contribution of our construction shows a useful class of applications for which 2PC protocols can be used to boost the security from static to adaptive in a *black-box* way. The well known and powerful "MPC-in-the-head" technique has found extensive applications in obtaining black-box construction of protocols that previously depended on generic Karp reductions. We believe that our technique yields an analogous "2PC-in-the-head" technique which might potentially be useful to obtain analogous constructions with adaptive security. In addition, we believe it will be useful in applications that rely on certain special properties of the Blum's Graph-Hamiltonicity ZK proof (BH). Concretely, we improve the Lindell-Zarosim adaptive ZK proof and the input-delayed protocol of Lapidot and Shamir [LS90] both of which relied on BH ZK-proof. More precisely, by relying on our ZK proof based on our instance-dependent commitment schemes that, in turn, depends on the NP relation in a black-box way, we save the cost of the expensive Karp reduction to Graph Hamiltonicity. We leave it as future work to determine if other applications that rely on the BH ZK-proof can be improved (e.g., NIZK).

2 Preliminaries

Basic notations. We denote the security parameter by κ. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is *negligible* if for every positive polynomial $p(\cdot)$ and all sufficiently large κ's it holds that $\mu(\kappa) < \frac{1}{p(\kappa)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. For an NP relation $\mathcal{R}$, we denote by $\mathcal{R}_x$ the

set of witnesses of x and by $\mathcal{L}_{\mathcal{R}}$ its associated language. That is, $\mathcal{R}_x = \{\omega \mid (x,\omega) \in \mathcal{R}\}$ and $\mathcal{L}_{\mathcal{R}} = \{x \mid \exists\, \omega \; s.t. \; (x,\omega) \in \mathcal{R}\}$.

We specify next the definitions of computationally indistinguishable and statistical distance.

Definition 2.1. *Let $X = \{X(a,\kappa)\}_{a\in\{0,1\}^*,\kappa\in\mathbb{N}}$ and $Y = \{Y(a,\kappa)\}_{a\in\{0,1\}^*,\kappa\in\mathbb{N}}$ be two distribution ensembles. We say that X and Y are* computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if for every* PPT *machine D, every $a \in \{0,1\}^*$, every positive polynomial $p(\cdot)$ and all sufficiently large κ's,*

$$\left|\Pr\left[D(X(a,\kappa),1^\kappa)=1\right] - \Pr\left[D(Y(a,\kappa),1^\kappa)=1\right]\right| < \frac{1}{p(\kappa)}.$$

Definition 2.2. *Let X_κ and Y_κ be random variables accepting values taken from a finite domain $\Omega \subseteq \{0,1\}^\kappa$. The* statistical distance *between X_κ and Y_κ is*

$$SD(X_\kappa, Y_\kappa) = \frac{1}{2}\sum_{\omega\in\Omega}\left|\Pr[X_\kappa=\omega] - \Pr[Y_\kappa=\omega]\right|.$$

We say that X_κ and Y_κ are ε-close *if their statistical distance is at most $SD(X_\kappa,Y_\kappa) \le \varepsilon(\kappa)$. We say that X_κ and Y_κ are* statistically close*, denoted $X_\kappa \approx_s Y_\kappa$, if $\varepsilon(\kappa)$ is negligible in κ.*

2.1 Commitment Schemes

Commitment schemes are used to enable a party, known as the *sender* S, to commit itself to a value while keeping it secret from the *receiver* R (this property is called *hiding*). Furthermore, in a later stage when the commitment is opened, it is guaranteed that the "opening" can yield only a single value determined in the committing phase (this property is called *binding*). In this work, we consider commitment schemes that are *statistically binding*, namely while the hiding property only holds against computationally bounded (non-uniform) adversaries, the binding property is required to hold against unbounded adversaries. Formally,

Definition 2.3 (Commitment schemes)**.** *A* PPT *machine $\mathsf{Com} = \langle S, R\rangle$ is said to be a non-interactive commitment scheme if the following two properties hold.*

Computational hiding: *For every (expected)* PPT *machine R^*, it holds that the following ensembles are computationally indistinguishable.*

- $\{\mathbf{View}^{\mathrm{R}^*}_{\mathsf{Com}}(m_1,z)\}_{\kappa\in\mathbb{N},m_1,m_2\in\{0,1\}^\kappa,z\in\{0,1\}^*}$
- $\{\mathbf{View}^{\mathrm{R}^*}_{\mathsf{Com}}(m_2,z)\}_{\kappa\in\mathbb{N},m_1,m_2\in\{0,1\}^\kappa,z\in\{0,1\}^*}$

where $\mathbf{View}^{R^*}_{\mathsf{Com}}(m,z)$ denotes the random variable describing the output of R^* after receiving a commitment to m using Com*.*

Statistical binding: *For any (computationally unbounded) malicious sender S^* and auxiliary input z, it holds that the probability that there exist valid decommitments to two different values for a view v, generated with an honest receiver while interacting with $\mathrm{S}^*(z)$ using* Com*, is negligible.*

We refer the reader to [Gol01] for more details. We recall that non-interactive perfectly binding commitment schemes can be constructed based on one-way permutations, whereas two-round statistically binding commitment schemes can be constructed based on one-way functions [Nao91]. We further consider *pseudorandom* commitments for which a honestly generated commitment to any message m is indistinguishable

from a uniform string of the same length. We note that such commitment schemes with statistical binding can be constructed based on one-way functions due to [Nao91] and with perfect binding based on one-way permutations. We conclude with the definition of trapdoor commitment schemes.

Definition 2.4 (Trapdoor commitment schemes)**.** *Let* $\mathsf{Com} = (\mathrm{S}, \mathrm{R})$ *be a statistically binding commitment scheme. We say that* Com *is a trapdoor commitment scheme is there exists an expected* PPT *oracle machine* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that for any* PPT R^* *and all* $m \in \{0,1\}^\kappa$*, the output* (τ, w) *of the following experiments is computationally indistinguishable:*

- *an honest sender* S *interacts with* R^* *to commit to* m*, and then opens the commitment:* τ *is the view of* R^* *in the commit phase, and* w *is the message* S *sends in the open phase.*
- *the simulator* $\mathcal{S}$ *generates a simulated view* τ *for the commit phase, and then opens the commitment to* m *in the open phase: formally* $(\tau, state) \leftarrow \mathcal{S}_1^{\mathrm{R}^*}(1^\kappa)$*,* $w \leftarrow \mathcal{S}_2(state, m)$*.*

2.2 Adaptive Instance-Dependent Commitment Schemes [LZ11]

We extend the instance-dependent commitment scheme definition of [LZ11], originally introduced for the binary message space, to a commitment scheme with an arbitrary message space $\mathcal{M}$.

Syntax. Let $\mathcal{R}$ be an NP relation and $\mathcal{L}$ be the language associated with $\mathcal{R}$. A (non-interactive) adaptive instance dependent commitment scheme (AIDCS) for $\mathcal{L}$ is a tuple of probabilistic polynomial-time algorithms $(\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$, where:

- Com is the commitment algorithm: For a message $m \in \mathcal{M}$, an instance $x \in \{0,1\}^*$ and a random string $r \in \{0,1\}^{p(|x|)}$ (where $p(\cdot)$ is a polynomial), $\mathsf{Com}(x, m; r)$ returns a commitment value c.
- Com' is a "fake" commitment algorithm: For an instance $x \in \{0,1\}^*$ and a random string $r \in \{0,1\}^{p(|x|)}$, $\mathsf{Com}'(x; r)$ returns a commitment value c.
- Adapt is an adaptive opening algorithm: Let $x \in \mathcal{L}$ and $\omega \in \mathcal{R}_x$. For all c and $r \in \{0,1\}^{p(|x|)}$ such that $\mathsf{Com}'(x; r) = c$, and for all $m \in \mathcal{M}$, $\mathsf{Adapt}(x, \omega, c, m, r)$ returns a pair (m, r') such that $c = \mathsf{Com}(x, m; r')$. (In other words, Adapt receives a "fake" commitment c and a message m, and provides an explanation for c as a commitment to the message m.

A decommitment to a commitment c is a pair (m, r) such that $c = \mathsf{Com}(x, m; r)$. Note the difference between Com and Com': Com is an ordinary committing algorithm (creating a commitment value for a given value), while for $x \in \mathcal{L}$ algorithm Com' creates commitment values that are not associated to any specific value. However, given a witness attesting to the fact that $x \in \mathcal{L}$, these commitments can later be claimed to be commitments to a specific m by using algorithm Adapt. We stress that without such a witness, a commitment generated by Com' cannot necessarily be decommitted to any value.

Security. We now define the notion of security for our commitment scheme.

Definition 2.5 (AIDCS)**.** *Let* $\mathcal{R}$ *be an* NP *relation and* $\mathcal{L} = \mathcal{L}_\mathcal{R}$*. We say that* $(\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$ *is a secure AIDCS for* $\mathcal{L}$ *if the following holds:*

1. *Computational hiding: For all* $m, m' \in \mathcal{M}$*, the ensembles* $\{\mathsf{Com}(x, m)\}_{x \in \mathcal{L}}$*,* $\{\mathsf{Com}(x, m')\}_{x \in \mathcal{L}}$ *and* $\{\mathsf{Com}'(x)\}_{x \in \mathcal{L}}$ *are computationally indistinguishable.*

2. *Adaptivity: For all $m \in \mathcal{M}$, the distributions $\{\mathsf{Com}(x, m; U_{p(|x|)}), m, U_{p(|x|)}\}_{x\in\mathcal{L},\omega\in\mathcal{R}_{\mathcal{L}}}$ and $\{\mathsf{Com}'(x; U_{p(|x|)}), m, \mathsf{Adapt}(x, \omega, \mathsf{Com}'(x; U_{p(|x|)}), m)\}_{x\in\mathcal{L},\omega\in\mathcal{R}_{\mathcal{L}}}$ are computationally indistinguishable (that is, the random coins that are generated by* Adapt *are indistinguishable from real random coins used by the committing algorithm* Com*).*

3. *Statistical binding: For all $m, m' \in \mathcal{M}$, $x \notin \mathcal{L}$ and a commitment c, the probability that there exist r, r' for which $c = \mathsf{Com}(x, m; r)$ and $c = \mathsf{Com}(x, m'; r')$ is negligible in κ.*

2.3 Interactive Hashing

An important sub-protocol that we rely on in most efficient instance-dependent commitment scheme is *interactive hashing*. We consider the following interactive hashing protocol of [NOVY98] that is executed between a sender S_{IH} and a receiver R_{IH} where the sender has private input $x \in \{0,1\}^{n_1}$ with length parameter 1^{n_1}, entropy threshold 1^{m_1} and a common input security parameter $1^{\kappa_{\mathrm{IH}}}$. It proceeds as follows:

1. For $i \in [m_1 + \kappa_{\mathrm{IH}}]$
   (a) R_{IH} selects a string $h_i \leftarrow 0^{i-1}1a_i$ where a_i is chosen uniformly from $\{0,1\}^{n_1-i}$.
   (b) S_{IH} sends $z_i = \langle h_i, x\rangle$ to R_{IH}.

We rely on the following proposition from [HRVW09] regarding the [NOVY98] protocol.

Proposition 2.1. *Let $L \subseteq \{0,1\}^{n_1}$ be a set of size at most 2^{m_1}. Let $\mathrm{S}^*_{\mathrm{IH}}$ be an (unbounded) adversary interactive with an honest receiver R_{IH}, then the following holds:*

$$\Pr[(\tau, (x_0, x_1), \cdot) \leftarrow (\mathrm{S}^*_{\mathrm{IH}}, \mathrm{R}_{\mathrm{IH}}) : x_0 \neq x_1 \ \wedge\ \forall\, b \in \{0,1\}\ x_b \in L \ \wedge\ x_b \text{ is consistent with } \tau] < 2^{-\Omega(\kappa_{\mathrm{IH}})}$$

where $(\tau, z_A, z_B) \leftarrow (A, B)$ means that τ is the transcript of the interaction between A and B and z_A and z_B are the respective outputs of A and B at the end of the execution. We say that x is consistent with a transcript with τ, if for every i $h_i(x) = z_i$, where h_i is the receiver's message and z_i is the sender's message in the i^{th} round.

2.4 Zero-knowledge Proofs

Definition 2.6 (Interactive proof system)**.** *A pair of* PPT *interactive machines $(\mathcal{P}, \mathcal{V})$ is called an* interactive proof system for a language $\mathcal{L}$ *if there exists a negligible function* negl *such that the following two conditions hold:*

1. COMPLETENESS: *For every $x \in \mathcal{L}$,*

$$\Pr[\langle \mathcal{P}, \mathcal{V}\rangle(x) = 1] \geq 1 - \mathsf{negl}(|x|).$$

2. SOUNDNESS: *For every $x \notin L$ and every interactive* PPT *machine B,*

$$\Pr[\langle B, \mathcal{V}\rangle(x) = 1] \leq \mathsf{negl}(|x|).$$

Definition 2.7 (Zero-knowledge)**.** *Let $(\mathcal{P}, \mathcal{V})$ be an interactive proof system for some language $\mathcal{L}$. We say that $(\mathcal{P}, \mathcal{V})$ is* computational zero-knowledge *if for every* PPT *interactive machine $\mathcal{V}^*$ there exists a* PPT *algorithm $\mathcal{S}$ such that*

$$\{\langle \mathcal{P}, \mathcal{V}^* \rangle(x)\}_{x \in \mathcal{L}} \overset{\text{c}}{\approx} \{\langle \mathcal{S} \rangle(x)\}_{x \in \mathcal{L}}$$

where the left term denote the output of $\mathcal{V}^*$ after it interacts with $\mathcal{P}$ on common input x whereas, the right term denote the output of $\mathcal{S}$ on x.

Definition 2.8 (Σ-protocol)**.** *A protocol π is a* Σ-protocol *for relation $\mathcal{R}$ if it is a 3-round public-coin protocol and the following requirements hold:*

- COMPLETENESS: *If $\mathcal{P}$ and $\mathcal{V}$ follow the protocol on input x and private input ω to $\mathcal{P}$ where $(x, \omega) \in \mathcal{R}$, then $\mathcal{V}$ always accepts.*
- SPECIAL SOUNDNESS: *There exists a polynomial-time algorithm A that given any x and any pair of accepting transcripts $(a, e, z), (a, e', z')$ on input x, where $e \neq e'$, outputs ω such that $(x, \omega) \in \mathcal{R}$.*
- SPECIAL HONEST-VERIFIER ZERO KNOWLEDGE: *There exists a* PPT *algorithm $\mathcal{S}$ such that*

  $$\{\langle \mathcal{P}(x, \omega), \mathcal{V}(x, e) \rangle\}_{x \in \mathcal{L}} \overset{\text{c}}{\approx} \{\mathcal{S}(x, e)\}_{x \in \mathcal{L}}$$

  where $\mathcal{S}(x, e)$ denotes the output of $\mathcal{S}$ upon input x and e, and $\langle \mathcal{P}(x, \omega), \mathcal{V}(x, e) \rangle$ denotes the output transcript of an execution between $\mathcal{P}$ and $\mathcal{V}$, where $\mathcal{P}$ has input (x, ω), $\mathcal{V}$ has input x, and $\mathcal{V}$'s random tape (determining its query) equals e.

2.5 Garbled Circuits

A core building block in our instance-dependent commitment schemes is garbled circuits, where these ideas are originated in [Yao86]. Here, a sender can encode a Boolean circuit that computes some PPT function f, in a way that (computationally) hides from the receiver any information but the function's output. Recalling that garbled circuits is an extremely useful tool for *statically* realizing any PPT function (for an arbitrary number of players), we note that it is insufficient for achieving security in the presence of static corruption of the receiver and adaptive corruption of the sender (when corruption takes place anytime during the protocol execution). This is due to the fact that the simulation of the corrupted receiver requires a simulator that produces a programmed fake garbling of the Boolean circuit that is always evaluated to the right output of the computed function, and only to that value. Nevertheless, if the party that produces the simulated garbling is adaptively corrupted, it must introduce randomness that is consistent with a correct generation of the garbling. Unfortunately, it is unknown how to complete the simulation for this case. Instead, we suggest to enhance the traditional view of garbled circuits with an additional algorithm that allows to capture the security property we need for our commitment protocols.

To be concrete, our notion of garbled circuits includes an additional algorithm of oblivious generation of a garbled circuit. Namely, given the randomness used to produce a garbled circuit $\widetilde{\mathrm{C}}$ of some circuit C, the algorithm generates new randomness that explains $\widetilde{\mathrm{C}}$ as the outcome of the simulated algorithm. We note that this modified notion of garbled circuits can be realized based on one-way functions, e.g., the construction from [LP09], for instance when the underlying symmetric key encryption used for garbling has an additional property of oblivious ciphertext generation (where a ciphertext can be sampled without the knowledge of the plaintext). Then the simulated garbling of a gate produces a garbled table using three obliviously generated ciphertexts and one ciphertext that encrypts the output label. We note that the ability

to switch from a standard garbled circuit to a simulated one will be exploited in our constructions below in order to equivocate a commitment to 0 into a commitment to 1.

Towards introducing our definition of garbled circuits we denote vectors by bold lower-case letters and use the parameter n to denote the input and output length for the Boolean circuit C.

Definition 2.9 (Garbling scheme)**.** *A garbling scheme* $\mathsf{Garb} = (\mathsf{Grb}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *consists of four polynomial-time algorithms that work as follows:*

- $(\widetilde{\mathrm{C}}, \mathbf{dk}, \mathsf{sk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C})$*: is a probabilistic algorithm that takes as input a circuit* C *with* $2n$ *input wires and* n *output wires and returns a garbled circuit* $\widetilde{\mathrm{C}}$*, a set of decoding keys* $\mathbf{dk} = (\mathrm{dk}_1, \ldots, \mathrm{dk}_n)$ *and a secret key* sk*.*

- $\widetilde{\mathbf{x}} := \mathsf{Enc}(\mathsf{sk}, \mathbf{x})$ *is a deterministic algorithm that takes an input a secret key* sk*, an input* $\mathbf{x}$ *and returns an encoded input* $\widetilde{\mathbf{x}}$*. We denote this algorithm by* $\widetilde{\mathbf{x}} := \mathsf{Enc}(\mathsf{sk}, \widetilde{\mathbf{x}})$*. In this work we consider* decomposable garbled schemes*. Namely, the algorithm takes multiple input bits* $\mathbf{x} = (x_1, \ldots, x_n)$*, runs* $\mathsf{Enc}(\mathsf{sk}, \cdot)$ *on each* x_i *and returns the garbled inputs* $\widetilde{x}_1$ *through* $\widetilde{x}_n$*, denoted by input labels.*

- $\widetilde{\mathbf{y}} := \mathsf{Eval}(\widetilde{\mathrm{C}}, \widetilde{\mathbf{x}})$*: is a deterministic algorithm that takes as input a garbled circuit* $\widetilde{\mathrm{C}}$ *and encoded inputs* $\widetilde{\mathbf{x}}$ *and returns encoded outputs* $\widetilde{\mathbf{y}}$*.*

- $\{\perp, y_i\} := \mathsf{Dec}(\mathrm{dk}_i, \widetilde{y}_i)$*: is a deterministic algorithm that takes as input a decoding key* dk_i *and an encoded output* $\widetilde{y}_i$ *and returns either the failure symbol* $\perp$ *or an output* y_i*. We write* $\{\perp, \mathbf{y}\} := \mathsf{Dec}(\mathbf{dk}, \widetilde{\mathbf{y}})$ *to denote the algorithm that takes multiple garbled outputs* $\widetilde{\mathbf{y}} = (\widetilde{y}_1 \ldots \widetilde{y}_n)$*, runs* $\mathsf{Dec}(\mathrm{dk}_i, \cdot)$ *on each* $\widetilde{y}_i$ *and returns the outputs* y_1 *through* y_n*.*

Correctness. We say that Garb is correct if for all $n \in \mathbb{N}$, for any polynomial-size circuit C, for all inputs $\mathbf{x}$ in the domain of C, for all $(\widetilde{\mathrm{C}}, \mathbf{dk}, \mathsf{sk})$ output by $\mathsf{Grb}(1^\kappa, \mathrm{C})$, for $\widetilde{\mathbf{x}} := \mathsf{Enc}(\mathsf{sk}, \mathbf{x})$ and $\widetilde{\mathbf{y}} := \mathsf{Eval}(\widetilde{\mathrm{C}}, \widetilde{\mathbf{x}})$ and for all $i \in [n]$, $y_i := \mathsf{Dec}(\mathrm{dk}_i, \widetilde{y}_i)$, where $(y_1, \ldots, y_n) = \mathrm{C}(\mathbf{x})$.

Security. We say that a garbling scheme Garb is secure if there exists a PPT algorithm SimGC such that for any polynomial-size circuit C, for all inputs $\mathbf{x}$ in the domain of C, for all $(\widetilde{\mathrm{C}}, \mathbf{dk}, \mathsf{sk})$ output by $\mathsf{Grb}(1^\kappa, \mathrm{C})$ and $\widetilde{\mathbf{x}} := \mathsf{Enc}(\mathsf{sk}, \mathbf{x})$ it holds that,

$$(\widetilde{\mathrm{C}}, \widetilde{\mathbf{x}}, \mathbf{dk}) \overset{\mathrm{c}}{\approx} \mathsf{SimGC}\,(1^\kappa, \mathrm{C}, \mathbf{y})$$

where $\mathbf{y} = \mathrm{C}(\mathbf{x})$.

Oblivious sampling. There exists a PPT algorithm OGrb such that for any polynomial-time circuit C and for all input/output pairs $(\mathbf{x}, \mathbf{y})$ such that $\mathrm{C}(\mathbf{x}) = \mathbf{y}$ it holds that,

$$\{r'_{\mathsf{Grb}}, \mathsf{SimGC}\left(1^\kappa, \mathrm{C}, \mathbf{y}; r'_{\mathsf{Grb}}\right)\}_{r'_{\mathsf{Grb}} \leftarrow \{0,1\}^*} \overset{\mathrm{c}}{\approx} \{\hat{r}_{\mathsf{Grb}}, \widetilde{\mathrm{C}}, \tilde{x}, \mathbf{dk}\}_{(\hat{r}_{\mathsf{Grb}}, \tilde{x}) \leftarrow \mathsf{OGrb}(1^\kappa, \mathrm{C}, \mathbf{x}, r_{\mathsf{Grb}})}$$

where r_{Grb} is the randomness for $(\widetilde{\mathrm{C}}, \mathbf{dk}, \mathsf{sk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C})$.

Note that correctness is perfect by our definition, which implies that a garbled circuit must be evaluated to the correct output. We further note that this notion is achieved by employing the point-and-permute optimization [PSSW09] to the garbling construction, as the evaluator of an honestly generated circuit always decrypts a single ciphertext for each gate which leads to the correct output. We further note that we assume that giving the secret key, it is possible to verify that the garbled circuit was honestly generated. Again, this holds with respect to existing garbling schemes, as the secret key includes the encoding of all input labels which allows to recompute the entire garbling and verifying the correctness of each gate.

2.6 Randomized Encoding

We review the definition of randomized encoding from [IK00, AIK04]. The following definition is produced almost verbatim from [AIK04].

Definition 2.10 (Randomized Encoding)**.** *Let $f : \{0,1\}^n \to \{0,1\}^\ell$ be a function. Then a function $\widehat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is said to be a* randomized encoding *of f, if:*

Correctness: *There exists a decoder algorithm B such that for any input $x \in \{0,1\}^n$, except with negligible probability over the randomness of the encoding and the random coins of B, it holds that $B(\widehat{f}(x, U_m)) = f(x)$.*

Computational (statistical) privacy: *There exists a* PPT *simulator $\mathcal{S}$, such that for any input $x \in \{0,1\}^n$ the following distributions are computationally (statistically) indistinguishable:*

- $\{\widehat{f}(x, U_m)\}_{n\in\mathbb{N}, x\in\{0,1\}^n}$,
- $\{\mathcal{S}(f(x))\}_{n\in\mathbb{N}, x\in\{0,1\}^n}$.

We require our randomized encoding to satisfy some additional properties:

1. **Robustness:** Applebaum et al. introduced in [AIKW13] the measures of *offline* and *online* complexities of an encoding, where the offline complexity refers to the number of bits in the output of $\widehat{f}(x, r)$ that solely depend on r and the online complexity refers to the number of bits that depend on both x and r. The motivation in their work was to construct *online efficient* randomized encoding, where the online complexity is close to the input size of the function. In our construction, we are not concerned specifically with the online complexity, but we require that there exists an offline part of the randomized encoding that additionally satisfies a robustness property. We present the definition of robustness for boolean functions f as it suffices for our construction.

   We say that $\widehat{f}$ is a *robust encoding* of f if there exist functions $\widehat{f}_{\text{OFF}}$ and $\widehat{f}_{\text{ON}}$ such that $\widehat{f}(x, r) = (\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r))$ and, in addition, it holds that: if there exists no x such that $f(x) = 1$, then for any r, there exists no z such that $B(\widehat{f}_{\text{OFF}}(r), z)$ outputs 1.

   Intuitively, robustness ensures that if the offline part was honestly computed using $\widehat{f}_{\text{OFF}}$ then there cannot exist any online part that can make the decoder output an element not in the range of the function f. We remark that it is possible to rewrite any randomized encoding as $(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r))$ for some functions $\widehat{f}_{\text{OFF}}$ and $\widehat{f}_{\text{ON}}$ (for instance, by setting $\widehat{f}_{\text{OFF}}$ to be the function that outputs the empty string and $\widehat{f}_{\text{ON}} = \widehat{f}$). Nevertheless, in order for the encoding to be robust there must exist a way to split the output bits of $\widehat{f}(x, r)$ into an offline part $\widehat{f}_{\text{OFF}}(r)$ and online part $\widehat{f}_{\text{ON}}(x, r)$ such that they additionally satisfy the robustness property. As mentioned before, it will not always be important for us to minimize the online complexity, where instead we require that the encoding is robust while minimizing the total (online+offline) complexity. We note that our definition is in the spirit of the authenticity definition with respect to garbled schemes from [BHR12].

2. **Oblivious sampling:** We require an additional oblivious property, as for the definition of garbling schemes, (that, looking ahead, will enable equivocation in our instance-dependence commitment schemes where a randomized encoding of function f can be explained as a simulated encoding). We denote this algorithm by ORE and define this new security property as follows.

For any function f as above and for all input/output pairs (x, y) such that $f(x) = y$ it holds that,

$$\{r', \mathcal{S}\left(y; r'\right)\}_{r' \leftarrow \{0,1\}^*} \overset{c}{\approx} \{r', \widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r)\}_{r' \leftarrow \mathsf{ORE}(x,r)}$$

where r is the randomness for generating $\widehat{f}$.

We note that garbling schemes meet our definition of robust randomized encoding. Specifically, the offline phase can be viewed as the garbled circuit and a commitment to the encoding of input x, whereas the online phase is the decommitment for the encoding of the input x. It is simple to verify that in case there exists no input x for which the circuit outputs 1, then a garbler cannot produce input labels for which the evaluation yields the encoding of 1. The oblivious sampling property directly follows if we rely on an encryption scheme that has the oblivious ciphertext generation property (which, in turn, can be easily obtained by viewing ciphertexts as obliviously sampled). Therefore, we have the following theorem:

Theorem 2.11. *Assuming the existence of one-way functions. Then, for any polynomial time computable boolean function $f : \{0,1\}^n \to \{0,1\}$, there exists a robust randomized encoding scheme $(\widehat{f}_{\text{OFF}}, \widehat{f}_{\text{ON}}, \mathcal{S})$ such that the offline complexity is $O(s\mathsf{poly}(\kappa))$ and online complexity is $O(n\mathsf{poly}(\kappa))$ where s is the size of the circuit computing f, n is the size of the input to f and κ is the security parameter.*

In Section 5, we show how to realize a robust randomized encoding scheme based on any two-party secure computation protocol (that meets certain requirements), which, in particular is satisfied by the [Yao86] and [GMW87] protocols. While this construction does not achieve any "non-trivial" online complexity, it will be sufficient for our application, as the total complexity will be $O(s\kappa)$.

2.7 Secure Multiparty Computation (MPC)

Our construction from Section 6 is inspired by the seminal work by Ishai et al. [IKOS09] that designs zero-knowledge proofs from secure multiparty protocols. Specifically, we first construct an instance-dependent commitment scheme and then combine it with their zero-knowledge protocol in order to achieve adaptive security. For that, we some of their definitions. Let n be the number of players, which will be denoted by $P_1, \ldots, P_n$. All players share the public input statement $x \in \mathcal{L}$ and a randomness share r_i. The *view* of P_i, denoted by V_i, includes x, r_i and the messages received by P_i during the execution of a protocol Π. Note that the messages sent by an uncorrupted player P_i as well as its local output can be inferred from V_i and x by invoking Π. The following definitions are taken from [IKOS09] verbatim.

Definition 2.12 (Consistent views)**.** *We say that a pair of views V_i, V_j are consistent (with respect to the protocol Π and some public input x) if the outgoing messages implicit in V_i are identical to the incoming messages reported in V_j and vice versa.*

We consider security of protocols in both the semi-honest and the malicious models. In the semi-honest model, one may break the security requirements into the following correctness and privacy requirements.

Definition 2.13 (Correctness)**.** *We say that Π realizes a deterministic n-party functionality $(x, r_1, \ldots, r_n)$ with perfect (resp., statistical) correctness if for all inputs $(x, r_1, \ldots, r_n)$, the probability that the output of some player is different from the output of f is 0 (resp., negligible in κ), where the probability is over the independent choices of the random inputs $r_1, \ldots, r_n$.*

Definition 2.14 (t-Privacy)**.** *Let $1 \leq t < n$. We say that Π realizes f with perfect t-privacy if there is a* PPT *simulator $\mathcal{S}$ such that for any inputs $(x, r_1, \ldots, r_n)$ and every set of corrupted players $T \subset [n]$, where $|T| \leq t$, the joint view $\mathbf{View}_T(x, r_1, \ldots, r_n)$ of players in T is distributed identically to $\mathcal{S}(T, x, \{r_i\}_{i \in T}, f_T(x, r_1, \ldots, r_n))$. The relaxations to statistical or computational privacy are defined in the natural way. That is, in the statistical (resp., computational) case we require that for every distinguisher D (resp., D with circuit size $poly(\kappa)$) there is a negligible function $\delta(\cdot)$ such that*

$$|Pr[D(\mathbf{View}_T(\kappa, x, r_1, \ldots, r_n)) = 1] - Pr[D(\mathcal{S}(\kappa, T, x, \{r_i\}_{i \in T}, f_T(x, r_1, \ldots, r_n))) = 1]| \leq \delta(\kappa).$$

In the malicious model, in which corrupted players may behave arbitrarily, security cannot be generally broken into correctness and privacy as above. However, for our purposes we only need the protocols to satisfy a weaker notion of security in the malicious model that is implied by the standard general definition. Specifically, it suffices that Π be t-private as defined above, and moreover it should satisfy the following notion of correctness in the malicious model.

Definition 2.15 (t-Robustness)**.** *We say that Π realizes f with perfect (resp., statistical) t-robustness if it is perfectly (resp., statistically) correct in the presence of a semi-honest adversary as in Definition 2.13, and furthermore for any computationally unbounded malicious adversary corrupting a set T of at most t players, and for any inputs $(x, r_1, \ldots, r_n)$, the following robustness property holds. If there is no $(x, r_1, \ldots, r_n)$ such that $f(x, r_1, \ldots, r_n) = 1$, then the probability that some uncorrupted player outputs 1 in an execution of Π in which the inputs of the honest players are consistent with $(x, r_1, \ldots, r_n)$ is 0 (resp., is negligible in κ).*

3 Static Zero-Knowledge Proofs from 2PC

Our technique also imply static ZK proofs from any two-party protocol that provides perfect correctness. Intuitively speaking, consider a two-party protocol that is corrupted in the presence of static adversaries with perfect correctness. Then, the prover generates the transcript of an execution where the parties' inputs are secret shares of the witness ω. That is, the parties' inputs are ω_1 and ω_2, respectively, such that $\omega = \omega_1 \oplus \omega_2$. Upon receiving a challenge bit from the verifier, the prover sends either the input and randomness of P_1 or P_2, for which the verifier checks for consistency with respect to the transcript, and that P_2 outputs 1. From the correctness of the underlying two-party protocol it holds that a malicious prover will not be able to answer both challenges, as that requires generating a complete accepting view. On the other hand, zero-knowledge is implied by the privacy of the two-party protocol.

We now proceed with the formal description of our zero-knowledge proof. Let x denote a statement in an NP language $\mathcal{L}$, associated with relation $\mathcal{R}$, let C be a circuit that outputs 1 on input (x, ω) only if $(x, \omega) \in \mathcal{R}$, and let $\Pi_g^{\mathrm{OT}} = \langle \pi_1, \pi_2 \rangle$ denote a two-party protocol that privately realizes C with perfect correctness; see Section 5 for the complete details of protocol Π_g^{OT} when embedded with our OT encoding. Our protocol is specified in Figure 1.

We note that our protocol implies the *first static zero-knowledge proof* based on (the two-party variant of) [GMW87] and [Yao86]. We next prove the following claim,

Theorem 3.1. *Assume the existence of one-way functions. Then, the protocol presented in Figure 1 is a static honest verifier zero-knowledge proof for any language in NP.*

Proof: Completeness follows easily from the fact that the honest prover knows the witness ω, thus it can answer both challenges of the verifier. On the other hand, from the perfect completeness of Π_g^{OT}, a malicious prover cannot provide randomness and input for both parties that are consistent with τ since that would

Static Zero-Knowledge Proof for any Language $\mathcal{L} \in \mathsf{NP}$

Inputs: A circuit C that computes the function $f(x, \omega) = \mathcal{R}(x, \omega)$ and a public statement $x \in \mathcal{L}$ for both. A witness ω for the validity of x for the prover $\mathcal{P}$.

The protocol:

1. $\mathcal{P} \rightarrow \mathcal{V}$: $\mathcal{P}$ invokes Π_g^{OT} and emulates the roles of P_1 and P_2 on random shares ω_1, ω_2 of ω, and randomness r_1, r_2. Let τ be the transcript of messages exchanged between these parties. $\mathcal{P}$ sends τ to the verifier.
2. $\mathcal{V} \rightarrow \mathcal{P}$: The verifier sends a random challenge bit $b \leftarrow \{0, 1\}$.
3. $\mathcal{P} \rightarrow \mathcal{V}$: Upon receiving the bit b the prover continues as follows,
   - If $b = 0$ then the prover sends (r_1, ω_1).
   - Else, if $b = 1$ then the prover sends (r_2, ω_2).
4. The verifier checks that the randomness and input are consistent with τ by emulating the corresponding party. In case of emulating P_2, the verifier checks that it further outputs 1.

Figure 1: Static zero-knowledge proof for any language $\mathcal{L} \in \mathsf{NP}$

imply that Π_g^{OT} computes an incorrect output for a false statement x that is not in $\mathcal{L}$, and violates the perfect correctness of Π_g^{OT}. Finally the zero-knowledge argument follows similarly to the proof of Theorem 5, where a simulated and real transcripts are indistinguishable. ■

Next, we extend this basic protocol to obtain an input-delayed zero-knowledge proof.

4 Instance-Dependent Commitments for the Binary Message Space

In the following two sections we discuss a general paradigm for designing adaptive instance-dependent commitments schemes for the binary message space, namely for the message space $\{0, 1\}$. Our constructions follow from two fundamental cryptographic primitives: garbling schemes (see Section 4.1) and robust randomized encoding (see Section 5.1.1), where the former can be viewed as a special case of the latter.

4.1 Instance-Dependent Commitments from Garbled Schemes

As a warmup, we present our first adaptive instance-dependent commitment scheme based on our garbled circuits notion as formally defined in Section 2.5 which, in turn, implies a construction for the binary message space $\{0, 1\}$ based on one-way functions (see more detailed discussion in Section 2.5). Let x denote a statement in an NP language $\mathcal{L}$, associated with relation $\mathcal{R}$, and let C be a circuit that outputs 1 on input (x, ω) only if $(x, \omega) \in \mathcal{R}$.[9] Intuitively speaking, our construction is described as follows.

A commitment to the bit 0 is defined by a garbling of circuit C , i.e., $\mathsf{Grb}(\mathrm{C})$, and a commitment to the secret key whereas a commitment to the bit 1 is defined by a simulated garbling of the circuit C with output set to 1, i.e., the garbled circuit output by $\mathsf{SimGC}(\mathrm{C}, 1)$, and a commitment the input encoding $\tilde{z}$ that is output by $\mathsf{SimGC}(\mathrm{C}, 1)$. The decommitment to the bit 0 requires revealing the secret key (all input labels)

[9] More explicitly, we assume that the common statement x is embedded inside the circuit and only ω is given as its input.

with which the receiver checks that $\mathsf{Grb}(\mathrm{C})$ is indeed a garbling of C. On the other hand, the decommitment to the bit 1 requires decommitting to $\tilde{z}$ with which the receiver checks that the simulated garbled circuit evaluates to 1. Importantly, if the committer knows a witness ω for the validity of x in $\mathcal{L}$, then it can always honestly commit to a garbling of circuit C and later decommit to both 0 and 1. For statements $x \in \mathcal{L}$, the hiding property of the commitment scheme follows directly from the indistinguishability of the simulated garbled circuit and the hiding property of the underlying commitment scheme. Whereas, for $x \notin \mathcal{L}$, the commitment is perfectly binding as even an unbounded committer cannot provide a honestly generated garbled circuit, and at the same time provide an encoding of some input that evaluates the garbled circuit to 1 (as there exists no witness ω for x). Finally, considering gabling constructions from the literature, such as the [LP09] scheme, we note that the communication complexity of our construction for committing a single bit equals $O(s\mathsf{poly}(\kappa))$ where s is the circuit's size and κ is the security parameter.

We are now ready to formally describe our construction in Figure 2. We further note that our construction can be based on one-way functions if we use the two-round Naor [Nao91] commitment scheme instead of a non-interactive commitment scheme based on one-way permutations. We prove the following theorem,

Instance-Dependent Commitment from Garbled Schemes

Building block: Let com denote a pseudorandom and statistically binding commitment scheme.

Inputs: Let circuit C be as above and let x denote a statement $x \in \mathcal{L}$.

The commitment scheme:

- $\mathsf{Com}(x, 0)$: S generates $(\widetilde{\mathrm{C}}, \mathbf{dk}, \mathsf{sk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C})$ and sends $\widetilde{\mathrm{C}}, \mathbf{dk}$ and $\sigma = \mathsf{com}(\mathsf{sk})$ to the receiver.

  Decommitment: S decommits σ to the secret key sk to the receiver, that verifies that C was garbled correctly and that decommitment information is correct.

- $\mathsf{Com}(x, 1)$: S generates $(\widetilde{\mathrm{SimC}}, \tilde{z}, \mathbf{dk}) \leftarrow \mathsf{SimGC}(1^\kappa, \mathrm{C})$ and sends $\widetilde{\mathrm{SimC}}, \mathbf{dk}$ and $\sigma = \mathsf{com}(\tilde{z}')$ to the receiver, where $\tilde{z}'$ is a complete set of input labels that involve $\tilde{z}$ and randomly chosen labels of the appropriate length.

  Decommitment: S decommits the encoding $\tilde{z}$ (and only that part within σ) to the receiver R, that computes $\tilde{y} := \mathsf{Eval}(\widetilde{\mathrm{SimC}}, \tilde{z})$ and then verifies whether $\mathsf{Dec}(\mathbf{dk}, \tilde{y})$ equals 1.

- $\mathsf{Com}'(x)$: S generates a commitment as for the case of $\mathsf{Com}(x, 0)$ using randomness r_{Grb}.

- $\mathsf{Adapt}(x, \omega, c, 0, r_{\mathsf{Grb}})$: If $\mathsf{Com}'(x; r_{\mathsf{Grb}}) \neq c$, then the algorithm returns $\perp$. Otherwise, it outputs the bit 0 and r_{Grb}.

- $\mathsf{Adapt}(x, \omega, c, 1, r_{\mathsf{Grb}})$: If $\mathsf{Com}'(x; r_{\mathsf{Grb}}) \neq c$, then the algorithm returns $\perp$. Otherwise, let $r_{\mathsf{Grb}} = (r_{\mathsf{Grb}}^{\mathsf{Garb}}, r_{\mathsf{Grb}}^{\mathsf{com}})$ denote the corresponding randomness used to generate the garbled circuit and σ, respectively. Then, the algorithm computes $\hat{r}_{\mathsf{Grb}} \leftarrow \mathsf{OGrb}(1^\kappa, \mathrm{C}, (x, \omega), r_{\mathsf{Grb}}^{\mathsf{Garb}})$ and returns the bit 1, $\hat{r}_{\mathsf{Grb}}$ and the randomness for explaining σ as a commitment of the encoding of ω as implied by sk, denoted by $\tilde{\omega}$.

Figure 2: Instance-dependent commitment from garbled schemes

Theorem 4.1. *Assume the existence of one-way functions. Then, the protocol presented in Figure 2 is a secure adaptive instance-dependent commitment scheme for any language in NP.*

Proof: The proof follows by demonstrating the three properties from Definition 2.5.

Computational hiding: Towards proving that, we need to show that the ensembles $\{\mathsf{Com}(x,0)\}_{x\in\mathcal{L}}$, $\{\mathsf{Com}(x,1)\}_{x\in\mathcal{L}}$ and $\{\mathsf{Com}'(x)\}_{x\in\mathcal{L}}$ are computationally indistinguishable. Note first that algorithm Com' is defined identically to $\mathsf{Com}(x,0)$, thus it is sufficient to prove that the ensembles $\{\mathsf{Com}(x,0)\}_{x\in\mathcal{L}}$ and $\{\mathsf{Com}(x,1)\}_{x\in\mathcal{L}}$ are computationally indistinguishable. Loosely speaking, this follows due to the indistinguishability of a garbled circuit from a simulated garbled circuit and the hiding property of the commitment scheme. In more details, recall that a commitment to 0 is a garbling of C and a commitment to sk, whereas a commitment to 1 is a simulated garbling of C and a commitment to $\tilde{z}'$. Moreover, a garbling of C is computationally indistinguishable from a simulated garbling of the same circuit by the security of garbling scheme. Whereas the hiding property of the commitment scheme com implies that a commitment to sk is indistinguishable from a commitment to $\tilde{z}'$. Combining the two arguments, and the fact that the committer does not need to reveal any information about the encoding of x, we define a hybrid commitment for which the circuit is garbled honestly (as in the case of committing to 0), yet the commitment to sk is replaced with a commitment to $\tilde{z}'$ (as in the case of committing to 1). We denote the distribution of this commitment scheme by $\{\mathsf{Com}_{\text{HYBRID}}\}$ and prove that

$$\{\widetilde{\mathrm{C}}, \mathbf{dk}, \mathsf{Com}(\mathsf{sk})\} \overset{\text{c}}{\approx} \{\mathsf{Com}_{\text{HYBRID}}\}$$

and

$$\{\mathsf{Com}_{\text{HYBRID}}\} \overset{\text{c}}{\approx} \{[\mathsf{SimGC}\,(1^\kappa, \mathrm{C}, \mathbf{y})]_1, [\mathsf{SimGC}\,(1^\kappa, \mathrm{C}, \mathbf{y})]_3, \sigma\}$$

where $\mathbf{y} = 1$ in our case and $[\mathsf{SimGC}\,(1^\kappa, \mathrm{C}, \mathbf{y})]_i$ denotes the i^{th} output of algorithm SimGC. The first indistinguishability proof is reduced to the hiding property of the commitment scheme, where a commitment to sk is indistinguishable from a commitment to $\tilde{z}'$. Thus, in the reduction an adversary that wishes to break this property, garbles the circuit C and associates this garbling with the an external string (that might be either be a commitment to sk or a commitment to $\tilde{z}'$). Finally, we claim that the second indistinguishability argument follows immediately from the security of the garbling scheme.

Adaptivity: Adaptivity follows from the fact that a "fake" commitment of 0, computed using algorithm Com', can be explained as a commitment to 1 by exploiting the obliviousness property of the garbling scheme. Namely, algorithm OGrb implies that it is possible to explain a garbled circuit generated by Grb as a simulated garbled circuit generated by SimGC. Moreover, com is a pseudorandom commitment. More formally, security is shown by constructing a simulator $\mathcal{S}_{\text{COM}}$ that produces the parties' views in the commitment phase, and then provides randomness that is consistent with the committer's message upon corruption. Specifically, the simulation of an honest committer is carried out by invoking algorithm $\mathsf{Com}'(x;r)$. Next, upon corrupting the committer, simulator $\mathcal{S}_{\text{COM}}$ obtains the committer's message m and $\omega \in \mathcal{R}_x$. If $m = 0$ then the simulator outputs r. Else, the simulator invokes algorithm $r' \leftarrow \mathsf{OGrb}(1^\kappa, \mathrm{C}, (x,\omega), r)$, and explains σ as a commitment to $\tilde{\omega}$, outputting randomness r' and the randomness for σ.

Finally, we need to prove that the following two distributions $\{\mathsf{Com}(x, m; U_{p(|x|)}), 1, U_{p(|x|)}\}_{x\in\mathcal{L},\omega\in\mathcal{R}_\mathcal{L}}$ and $\{\mathsf{Com}'(x; U_{p(|x|)}), 1, \mathsf{Adapt}(x, \omega, \mathsf{Com}'(x; U_{p(|x|)}))\}_{x\in\mathcal{L},\omega\in\mathcal{R}_\mathcal{L}}$ are computationally indistinguishable, which follows from the oblivious sampling of the garbled circuit and the pseudorandomness of the commitment scheme com. Namely, the first distribution corresponds to a honest commitment of 1 which yields $(\widetilde{\mathrm{SimC}}, \mathbf{dk}, r, \sigma)$, whereas the second distribution corresponds to an execution by the oblivious sampler which yields $(\widetilde{\mathrm{C}}, \mathbf{dk}, r', \sigma)$. By the oblivious sampling property specified in Section 2.5, the first three items within the two distributions are computationally indistinguishable. Moreover, σ is indistinguishable in both

distributions due to the hiding property of com. A formal statement follows using a hybrid argument as explained above.

Perfect binding: Finally, for an invalid statement x that is not in $\mathcal{L}_{\mathcal{R}}$, binding is ensured by the perfect correctness property of the garbling scheme and the fact that for a false statement there exists no input for C for which the circuit is evaluated to 1. Thus, a committer cannot commit to 0 by producing a real garbled circuit and then decommit to 1, and vice versa. More formally, let $(\widetilde{\mathrm{C}}, \mathbf{dk}, \sigma)$ denote a commitment to 0 as specified in Figure 2. Then $(\widetilde{\mathrm{C}}, \mathbf{dk}, \sigma)$ cannot be decommitted into 1 as that requires specifying a garbled input z' for algorithm Eval for which $\widetilde{\mathrm{C}}$ is evaluated to 1. Nevertheless, since there exists no such input then equivocation to 1 is not possible. Moreover, if the commitment is comprised from $(\widetilde{\mathrm{SimC}}, \mathbf{dk}, \sigma)$, then a dishonest committer cannot decommit it into 0 as that implies that it has an encoding for some input that evaluates the real garbled circuit to 1. By the correctness of the garbling, such an encoding does not exist. ■

4.2 On Obtaining Instance-Dependent Trapdoor Commitment Schemes

As a side note, we observe that our construction implies instance-dependent trapdoor commitment scheme where the secret trapdoor of the construction is the witness. To see that, consider a standard garbling construction without the additional obliviousness property that we require in Definition 2.9. Moreover, consider the same commitment/decommitment algorithms for both 0 and 1 as specified in Figure 2. Then, it is simple to verify that computational hiding and perfect binding hold as above with respect to the validity of the proven statement x. This is because non of these properties is implied by the additional obliviousness property. Finally, we note that a committer who holds the witness ω, can first commit to 0 and then later equivocate its commitment by revealing the encoding of (x, ω) (which amounts to a decommiment to 1 as such an encoding evaluates the garbled circuit to 1). We stress that the witness should not need to be given to the committer prior to the commitment phase in order to achieve equivocation. This implies the following,

Theorem 4.2. *Assume the existence of one-way functions. Then, the there exists a protocol that is a secure instance-dependent trapdoor commitment scheme for any language in NP.*

Note that our construction improves over prior work for which instance-dependent trapdoor commitment schemes were only known for Σ-protocols [Dam10].

5 Randomized Encoding from Two-Party Computation

In this section, we show how to construct a randomized encoding for any function f, given a two-party computation in the oblivious transfer (OT)-hybrid. This is opposed to prior works that have established the usefulness of randomized encoding in constructing efficient multiparty computation [IK00, AIK04, DI06].

Let $f : \{0,1\}^n \to \{0,1\}$ be an arbitrary polynomial-time computable function. We define $g(a, b) = f(a \oplus b)$ and view g as a two-party functionality. Then let $\Pi_g^{\mathrm{OT}} = \langle \pi_1, \pi_2 \rangle$ be a two-party protocol which realizes g with the following guarantees:

1. It guarantees UC security against semi-honest adversaries in the OT-hybrid that can statically corrupt either P_1 or P_2 and adaptively corrupt P_2. Looking ahead, we consider two different adversaries: (1) adversary $\mathcal{A}_1$ that corrupts P_1 at the beginning of the execution and adaptively corrupts P_2 post-execution (further denoted as a semi-adaptive adversary [GWZ09]) and (2) adversary $\mathcal{A}_2$ that corrupts P_2 at the beginning of the execution. We denote the corresponding simulators by $\mathcal{S}_1$ and $\mathcal{S}_2$.

2. Finally, we require that P_1 is the (designated) sender for all OT instances and that the output of the computation is obtained only by P_2.

We remark that both the classic Yao's garbled circuit construction [Yao86] and the [GMW87] protocol satisfy these conditions in the OT-hybrid. We further stress that while garbled circuit constructions do not (in general) admit adaptive corruptions, we show that the specific corruption by adversary $\mathcal{A}_1$ can be simulated in the OT-hybrid. In Section 5.2 we discuss these two realizations in more details. We next demonstrate how to transform any two-party computation protocol that satisfies the properties listed above to a randomized encoding. Our first construction will rely on trapdoor permutations to realize the OT functionality. We then relax this requirement and show how to rely on one-way functions.

Given any protocol Π_g^{OT} we consider a protocol $\widetilde{\Pi}$ that is obtained from Π_g^{OT} by replacing every OT call with the enhanced trapdoor permutation based OT protocol of [EGL85]. Let $\{f_{\mathrm{TDP}} : \{0,1\}^n \to \{0,1\}^n\}$ be a family of trapdoor permutations and h be the corresponding hard-core predicate. More precisely,

- For every OT call where P_1's input is (s_0, s_1) and P_2's input is t, we require P_1 to send the index of a trapdoor permutation f_{TDP} to P_2. Next, P_2 samples v_{1-t} and u_t uniformly at random from $\{0,1\}^n$ and sets $v_t = f_{\mathrm{TDP}}(u_t)$. P_2 sends (v_0, v_1) to P_1, that is followed by the message (c_0, c_1) from P_1 to P_2 where $c_0 = h(u_0) \oplus s_0$ and $c_1 = h(u_1) \oplus s_1$ and $u_0 = f_{\mathrm{TDP}}^{-1}(v_0), u_1 = f_{\mathrm{TDP}}^{-1}(v_1)$.

We need to verify that $\widetilde{\Pi}$ satisfies all the required properties.

1. It follows from the fact that if Π_g^{OT} implements g with UC security against semi-honest adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, then $\widetilde{\Pi}$ achieves the same against corresponding adversaries that corrupt the same parties and finally output the view of P_2. In more detail, recall that $\mathcal{A}_1$ corrupts P_1 at the beginning and P_2 post execution (adaptively). Now, since Π_g^{OT} admits simulation of $\mathcal{A}_1$ in the OT-hybrid, for the same property to hold for $\widetilde{\Pi}$, it suffices to achieve simulation of the OT protocol where the sender is corrupted at the beginning and the receiver is corrupted post execution. It is easy to see that the [EGL85] protocol satisfies this requirement since the receiver is equivocable. Next, to see that $\mathcal{A}_2$ can be simulated we rely on the fact that the OT protocol described above admits (semi-honest) receiver's simulation. Therefore, $\widetilde{\Pi}$ satisfies all the required properties.

2. This property directly holds as we rely on the same instructions to determine the sender and receiver of the OT calls.

Our randomized encoding. We now proceed with the description of our robust randomized encoding of f as formalized in Definition 2.10 by specifying the functions $\widehat{f}_{\mathrm{OFF}}$, $\widehat{f}_{\mathrm{ON}}$ and the simulation $\mathcal{S}$.

Towards describing our algorithms, we consider a real world experiment carried out between parties P_1 and P_2 that engage in an execution of $\widetilde{\Pi}$ with environment $\mathcal{Z}$. Let $\mathbf{REAL}_{\widetilde{\Pi},\mathcal{A},\mathcal{Z}}(\kappa, x, \mathbf{r})$ denote the output of $\mathcal{Z}$ on input x, random tape $r_{\mathcal{Z}}$ and a security parameter κ upon interacting with $\mathcal{A}$ with random tape $r_{\mathcal{A}}$ and parties P_1, P_2 with random tapes r_1, r_2, respectively, that engage in protocol $\widetilde{\Pi}$ where the inputs are determined by $\mathcal{Z}$ and $\mathbf{r} = (r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, r_2)$. Let $\mathbf{REAL}_{\widetilde{\Pi},\mathcal{A},\mathcal{Z}}(\kappa, x)$ denote a random variable describing $\mathbf{REAL}_{\widetilde{\Pi},\mathcal{A},\mathcal{Z}}(\kappa, x, \mathbf{r})$ where the random tapes are chosen uniformly. We denote by $\mathbf{IDEAL}_{g,\mathcal{S},\mathcal{Z}}(\kappa, x, \mathbf{r})$ the output of $\mathcal{Z}$ on input x, random tape $r_{\mathcal{Z}}$ and security parameter κ upon interacting with $\mathcal{S}$ and parties P_1, P_2, running an ideal process with random tape $r_{\mathcal{S}}$, where $\mathbf{r} = (r_{\mathcal{Z}}, r_{\mathcal{S}})$. Let $\mathbf{IDEAL}_{g,\mathcal{S},\mathcal{Z}}(\kappa, x)$ denote a random variable describing $\mathbf{IDEAL}_{g,\mathcal{S},\mathcal{Z}}(\kappa, x, \mathbf{r})$ when the random tapes $r_{\mathcal{Z}}$ and $r_{\mathcal{S}}$ are chosen uniformly.

Encoding: Consider a (semi-honest) adversary $\mathcal{A}_1$ that corrupts P_1 at the beginning of the execution. At the end of the execution, $\mathcal{A}_1$ first sends τ to $\mathcal{Z}$ where τ is the transcript of messages exchanged between

P_1 and P_2. Next it (adaptively) corrupts P_2 and sends (a_2, r_2) to $\mathcal{Z}$ where a_2 and r_2 are the respective input and randomness used by party P_2. Let $\mathcal{S}_1$ be the corresponding simulator as guaranteed by the properties of $\widetilde{\Pi}$.

1. $\widehat{f}_{\text{OFF}}(r)$: The offline encoding is obtained by running $\mathcal{S}_1$ with randomness $r_{\mathcal{S}_1}$ until it sends the first message to the environment. Recall that $\mathcal{S}_1$ statically corrupts P_1, where upon completing the execution, $\mathcal{S}_1$ sends the transcript of the messages to the environment. We define the output of $\widehat{f}_{\text{OFF}}(r)$ to be this output where the input a_1 of party P_1 is sampled uniformly at random. Notice that the offline part of the encoding does not depend on the input x as required.
2. $\widehat{f}_{\text{ON}}(x, r)$: To obtain the online part, we continue the execution of $\mathcal{S}_1$ in the execution corresponding to the transcript τ generated by $\widehat{f}_{\text{OFF}}(r)$. Recall that after sending τ, $\mathcal{S}_1$ adaptively corrupts P_2 and sends the input and random tape of P_2 to the environment. $\widehat{f}_{\text{ON}}(x, r)$ continues the emulation of $\mathcal{S}_1$, where upon corrupting party P_2 it feeds $\mathcal{S}_1$ with the input of P_2 as $a_2 = x \oplus a_1$ and $f(x)$ as the output. The simulation returns the view of P_2 and $\widehat{f}_{\text{ON}}(x, r)$ is set to (a_2, r_2) where r_2 is the random tape of P_2 output by $\mathcal{S}_1$.

Decoder: The decoder B on input $(z_{\text{OFF}}, z_{\text{ON}})$ recomputes the view of P_2 from the messages sent by P_1 to P_2 in z_{OFF} and the input and randomness of P_2 in z_{ON}. It checks if the messages sent from P_2 to P_1 are consistent with what is in z_{OFF} and finally outputs what P_2 outputs in the execution.

Simulation: Consider the (semi-honest) adversary $\mathcal{A}_2$ that statically corrupts P_2. At the end of the execution $\mathcal{A}_2$ sends $(\tau, (a_2, r_2))$ to $\mathcal{Z}$ where τ is the transcript of messages exchanged between P_1 and P_2 and a_2 and r_2 are the respective input and randomness used by party P_2. Let $\mathcal{S}_2$ be the corresponding simulator. Then the simulation algorithm of the randomized encoding $\mathcal{S}$ is defined as follows. Upon receiving $y = f(x)$, $\mathcal{S}$ invokes $\mathcal{S}_2$ where P_2's input is set to a uniformly chosen random string a_2 and its output is set to y. Recall that $\mathcal{S}_2$ outputs $(\tau, (a_2, r_2))$ at the end of the execution. Then the output of $\mathcal{S}$ is defined by $(s_{\text{OFF}}, s_{\text{ON}})$ where $s_{\text{OFF}} = \tau$ and $s_{\text{ON}} = (a_2, r_2)$.

We next prove the following theorem.

Theorem 5.1. *Let $(\widehat{f}(x, r), \mathcal{S}, B)$ be as above. Then $\widehat{f}(x, r)$ is a randomized encoding of f with computational privacy. Assuming the existence of enhanced trapdoor permutations, we obtain an encoding with offline complexity $C_\Pi + \rho_\Pi \kappa$ and online complexity $|x| + r_\Pi + \rho_\Pi \kappa$ where C_Π is the communication complexity of Π_g^{OT} in the OT-hybrid, ρ_Π in the number of OT invocations made by P_2, r_Π is the randomness complexity of P_2 in Π_g^{OT} and κ is the security parameter. If we instead rely on one-way functions we achieve an encoding with offline and online complexities $C_\Pi + \rho_\Pi \mathsf{poly}(\kappa)$ and $|x| + r_\Pi + \rho_\Pi \mathsf{poly}(\kappa)$, respectively.*

Proof: We continue with the arguments of the two properties required for our randomized encoding: correctness and privacy. As the correctness argument relies on an argument made in the proof for claiming privacy, we start with the privacy proof. Towards this, we will consider a specific environment $\mathcal{Z}^*$ that assigns inputs to the parties as follows. $\mathcal{Z}^*$ gives P_1 and P_2 inputs a_1 and a_2 where a_1 is chosen at random and $a_2 = a_1 \oplus x$. At the end of the execution $\mathcal{Z}^*$ outputs all messages received from $\mathcal{A}$ as its output.

Privacy. We prove the indistinguishability of a real and a simulated encoding. At first glance, it may seem that the real encoding and the simulated encoding are quite different as they rely on the simulation of different adversaries. We begin with observation that the joint distribution of $(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r))$ is identically distributed to the ideal execution $\mathbf{IDEAL}_{g, \mathcal{S}_1, \mathcal{Z}^*}(\kappa, x)$. This is because the distribution of inputs

and outputs provided for P_1 and P_2 by the encoding algorithm is identical to the distribution of inputs and outputs assigned by $\mathcal{Z}^*$. Analogously, it follows that the distribution of the simulated encoding generated by $\mathcal{S}$ is the identically distributed to $\mathbf{IDEAL}_{g,\mathcal{S}_2,\mathcal{Z}}(\kappa, x)$. More precisely,

$$(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x,r)) \equiv \mathbf{IDEAL}_{g,\mathcal{S}_1,\mathcal{Z}^*}(\kappa, x), \text{ whereas}$$
$$\mathcal{S}(f(x)) \equiv \mathbf{IDEAL}_{g,\mathcal{S}_2,\mathcal{Z}^*}(\kappa, x).$$

We prove indistinguishability via a standard hybrid argument. First, it follows from the indistinguishability of the simulations generated by $\mathcal{S}_1$ and $\mathcal{S}_2$ that:

$$\mathbf{IDEAL}_{g,\mathcal{S}_1,\mathcal{Z}^*}(\kappa, x) \overset{\text{c}}{\approx} \mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_1,\mathcal{Z}^*}(\kappa, x), \text{ and}$$
$$\mathbf{IDEAL}_{g,\mathcal{S}_2,\mathcal{Z}^*}(\kappa, x) \overset{\text{c}}{\approx} \mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_2,\mathcal{Z}^*}(\kappa, x).$$

Recall that both adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ send $(\tau, (a_2, r_2))$ where τ is the transcript of messages exchanged between the parties and a_2 and r_2 are the respective input and randomness of P_2. Furthermore, from the description of our environment $\mathcal{Z}^*$, we know that $\mathcal{Z}^*$ simply outputs whatever it receives from the adversary. Now, as the adversaries are semi-honest and send identical information to $\mathcal{Z}^*$, we conclude the proof of indistinguishability by observing that

$$\mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_1,\mathcal{Z}^*}(\kappa, x) \equiv \mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_2,\mathcal{Z}^*}(\kappa, x).$$

Therefore, indistinguishability is implied.

Correctness. We need to show that for every x, $B(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x,r))$ outputs $f(x)$ except with negligible probability, where the probability is over the choices of r and the random coins of B. Due to the facts that $\mathbf{IDEAL}_{g,\mathcal{S}_1,\mathcal{Z}^*}(\kappa, x) \overset{\text{c}}{\approx} \mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_1,\mathcal{Z}^*}(\kappa, x)$ and that P_2 outputs $f(x)$ except with negligible probability in the real world experiment, it follows that P_2 must output $f(x)$ except with negligible probability relative to its view output in the ideal world. Therefore, B must output $f(x)$ with the same probability as well. If we additionally require perfect correctness, then we need to assume that protocol $\widetilde{\Pi}$ is perfectly correct and the simulation generated by $\mathcal{S}_1$ is perfect. ■

Relaxing to one-way functions. Recall that in the preceding construction we relied on enhanced trapdoor permutation family for realizing the OT-hybrid. We now argue that the same can be accomplished using only one-way functions. Towards this, we first observe that if we use a one-way permutation (OWP) as opposed to a trapdoor permutation in protocol $\widetilde{\Pi}$, then our construction still satisfies all the required properties. This is because we do not require the protocol instantiated with OWP to be played by two efficient parties P_1 and P_2. Specifically, it suffices to have the simulations generated by $\mathcal{S}_1$ and $\mathcal{S}_2$ be efficient and indistinguishable. This is because the simulators are not required to use the trapdoor in order to simulate the OT calls in our instantiation based on trapdoor permutations, as they can ensure that they know the preimages of all the image elements in the transcript.

Finally, to relax to one-way functions, we realize the OT calls by relying on a randomized encoding of OT defined as follows:

- For every OT call where P_1's input is (s_0, s_1) and P_2's input is t, we incorporate it in the encoding by generating a transcript containing the commitments c_0 and c_1 of s_0 and s_1 using a statistically binding

commitment scheme com (which can be based on one-way functions), placing the decommitment information of c_t in P_2's random tape.[10]

Note that this OT encoding does admit static corruption of P_1 followed by adaptive corruption of P_2 as we can honestly generate the commitments using P_1's inputs and place the decommitment information appropriately in P_2's random tape when it is corrupted post execution. This encoding also admits static corruption of P_2 as we simply commit to the required value s_t, use a random value in place of s_{1-t}, and place the decommitment of s_t in P_2's random tape.

Complexity. Finally, we measure the complexity of our encoding. Note first that for each OT call the offline encoding is a pair of image elements of the one-way permutation incurring $O(\kappa)$ overhead, while the online complexity is a preimage of length κ. Then the offline encoding of the overall construction is the communication complexity of $\widetilde{\Pi}$ which equals to the communication of Π_g^{OT}, denoted by C_Π, together with the number of OT calls, denoted by ρ_Π, which overall yields $C_\Pi + \rho_\Pi O(\kappa)$. Moreover, the online encoding includes P_2's input a_2 and randomness r_2 where the latter includes the randomness complexity of Π_g^{OT} and the complexity of the receiver's randomness for the OT invocations which is $|x| + r_\Pi + \rho_\Pi \kappa$. If we rely on one-way functions then the OT calls are incorporated as commitments and incur $\mathsf{poly}(\kappa)$ per invocation for the commitment as well as the decommitment algorithms.

5.1 Corollaries and Applications

In this section, we demonstrate the power of the proceeding transformation by proving lower bounds and providing additional applications.

5.1.1 Instance-Dependent Commitment Schemes

In the following, we require two additional prosperities on our randomized encoding to construct an instance-dependent commitment. These are *robustness* and *oblivious sampling*. Robustness requires that for an honestly generated encoding of the offline part there exists no string that serves as the online part that makes the decoder output an element outside the range of f. On the other hand, oblivious sampling requires that given any (honestly generated) encoding it is possible to provide randomness for the simulation that produces the same encoding. We show that each of the two properties can be guaranteed relative to the construction specified in Section 5 if the underlying secure two-party protocol Π_g^{OT} satisfies some additional properties.

Robustness. To achieve robustness we require that Π_g^{OT} satisfies the following additional property:

- We say that a two-party computation protocol is *robust* if the following holds: Suppose there exists no x such that $f(x) = 1$, then in any execution using Π_g^{OT} where both P_1 and P_2 act honestly (potentially using a maliciously generated random tape), P_2 can never output 1.

We remark that this definition of robustness is slightly different from the notion of t-robustness defined for MPC in Section 2.7. In the definition presented above for two-party computation, we require that it holds against an adversary that is only allowed to (maliciously) determine the random tapes of both parties who later interact honestly according to the protocol specification. In contrast, the definition for MPC as required in [IKOS09] requires that the property holds against an adversary that in addition to (maliciously)

[10] Note that, in Naor's statistically binding commitment scheme the decommitment information is the inverse under a pseudorandom generator that is uniformly sampled, and hence can be placed in the random tape.

determining the random tapes of all honest parties, can corrupt t parties and cause it to arbitrarily deviate from the protocol specification. Our variant for the two-party protocol is similar in spirit to the notion of *semi-malicious* security defined in [BGJ+13]. As noted in [BGJ+13] the protocol of [GMW87] satisfies this definition.[11] In fact, it suffices for the underlying OT protocol to statistically bind the sender's input to the transcript. Hence, this implies that our modified protocol $\widetilde{\Pi}$ where we realize OT using the [EGL85] protocol satisfies robustness.

Given this property, we can establish the robustness of our randomized encoding. Specifically, observe that the offline part of the actual encoding $\widehat{f}_{\text{OFF}}(x, r)$, i.e., the transcript of the exchanged messages is generated by using the randomness r to provide input and randomness to P_1 consistent with the transcript. Furthermore, by the way we encode the oblivious transfer, it binds the sender's inputs to the oblivious transfer executions. On the other hand, the online part produces the corresponding input and randomness for P_2. Thus for an offline part τ and a valid randomness r such that $\widehat{f}_{\text{OFF}}(x, r) = \tau$ if there is a corresponding online part, then it holds that there exists an execution between parties P_1 and P_2 with some inputs and (potentially maliciously chosen) randomness that results in the transcript τ. Therefore from the robustness of the underlying two-party protocol it follows that P_2 cannot output 1, as 1 is not in the range of f which, in turn, means that 1 is not in the range of g.

In order to relax our construction to one-way functions using statistically binding commitments, the robustness property must require that a commitment cannot be equivocated as this will allow the sender of the OT to commit to one value and the receiver to decommit to another. Unfortunately, statistically binding commitments can be equivocated with maliciously generated randomness. In order to overcome this problem, we modify the definition of robustness which will be sufficient for our application of instance-dependence commitments. Namely, we require that there is a public-parameter pp for which the encoding function takes as input such that robustness holds with high probability over a randomly generated pp. More formally, we say that $\widehat{f}$ is a *statistically robust encoding* of f if there exist functions $\widehat{f}_{\text{OFF}}$ and $\widehat{f}_{\text{ON}}$ such that $\widehat{f}(pp, x, r) = (pp, \widehat{f}_{\text{OFF}}(pp, r), \widehat{f}_{\text{ON}}(pp, x, r))$ and, in addition, it holds that: if there exists no x such that $f(x) = 1$, then for any r the probability over pp that there exists τ such that $B(\widehat{f}_{\text{OFF}}(r), \tau) = 1$ is negligible.

Now, we can incorporate our OT calls using statistically binding commitments (based on one-way functions) and require pp to be the set of the first messages required for all the commitments used in the encoding.

Oblivious sampling. To obtain this property, we will require that Π_g^{OT} satisfy that:

- Any view of P_2 in an execution of Π_g^{OT} can be explained as the output of a simulation of an adversary that statically corrupts P_2. This property is required for the Adapt algorithm of our instance-dependent commitment scheme. We formalize this by requiring an algorithm InvSamp that is giving as input a view of P_2 denoted by $(\tau, (a_2, r_2))$ where τ is the transcript of messages exchanged and a_2 and r_2 are the respective input and random tape of P_2, and outputs random coins r' for $\mathcal{S}_2$ such that $\mathcal{S}_2$ on input $(a_2, f(x))$ and random tape r' outputs the view $(\tau, (a_2, r_2))$. Furthermore, we require that InvSamp on input a view generated by $\mathcal{S}_2$ with random tape r' returns the same random tape r'.

Recall that in our construction we rely on $\widetilde{\Pi}$ which is exactly the protocol Π_g^{OT} with the exception that the OT calls are replaced using the protocol of [EGL85]. If Π_g^{OT} has an InvSamp algorithm, it is possible to extend this algorithm to obtain an analogous algorithm for $\widetilde{\Pi}$ as we can combine InvSamp with an algorithm that achieves the same for the OT subprotocol implemented using [EGL85], where the latter follows due to

[11] If we assume encryption scheme with zero decryption error, then even [Yao86]'s protocol satisfies robustness.

the equivocality of the receiver.[12] Hereafter, when we refer to $\mathsf{InvSamp}$ we will refer to the algorithm that corresponds to protocol $\widetilde{\Pi}$.

We recall that the simulator for the encoding $\mathcal{S}$ uses its random tape to sample P_2's input a_2 and the random tape of $\mathcal{S}_2$ that it executes internally. We identify the random tape of $\mathcal{S}$ as (a_2, r') where a_2 will be used for P_2's input and r' will be the random tape of $\mathcal{S}_2$ in the internal emulation by $\mathcal{S}$. This means that an execution of $\mathcal{S}(f(x))$ with random tape r^* can be rewritten as $\{\mathcal{S}_2(a_2, f(x); r')\}$ where $r^* = (a_2, r')$. Furthermore, from the properties of our $\mathsf{InvSamp}$ algorithm we have that

$$\mathsf{InvSamp}(\mathcal{S}_2(a_2, f(x); r')) = r'.$$

Next, we describe the oblivious sampling algorithm for our randomized encoding that allows to demonstrate an actual encoding as a simulated encoding. Upon receiving an encoding, the $\mathsf{OblSamp}$ algorithm needs to produce a random tape for $\mathcal{S}$ that generates the same encoding. That is, upon receiving as input an encoding $(\tau, (a_2, r_2))$, $\mathsf{OblSamp}$ outputs $(a_2, \mathsf{InvSamp}(\tau, (a_2, r_2)))$. It follows from the construction that when $\mathcal{S}$ is executed with random tape $(a_2, \mathsf{InvSamp}(\tau, (a_2, r_2)))$ it will output $(\tau, (a_2, r_2))$.

Next, from privacy property of our encoding we have that

$$(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r)) \stackrel{\text{c}}{\approx} \mathcal{S}(f(x)).$$

This means that

$$\begin{aligned}
&\{(\tau, (a_2, r_2)) \leftarrow (\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r)) : \mathsf{OblSamp}(\tau, (a_2, r_2))\} \\
&\quad \equiv \{(\tau, (a_2, r_2)) \leftarrow (\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r)) : (a_2, \mathsf{InvSamp}(\tau, (a_2, r_2)))\} \\
&\quad \stackrel{\text{c}}{\approx} \{(\tau, (a_2, r_2)) \leftarrow \mathcal{S}(f(x)) : (a_2, \mathsf{InvSamp}(\tau, (a_2, r_2)))\} \\
&\quad \equiv \{a_2 \leftarrow \{0,1\}^n; r' \leftarrow \{0,1\}^*; (\tau, (a_2, r_2)) \leftarrow \mathcal{S}_2(a_2, f(x); r') : (a_2, \mathsf{InvSamp}(\tau, (a_2, r_2)))\} \\
&\quad \equiv \{a_2 \leftarrow \{0,1\}^n; r' \leftarrow \{0,1\}^*; (\tau, (a_2, r_2)) \leftarrow \mathcal{S}_2(a_2, f(x); r') : (a_2, r')\} \\
&\quad \equiv \{a_2 \leftarrow \{0,1\}^n; r' \leftarrow \{0,1\}^* : (a_2, r')\}.
\end{aligned}$$

Now, since $\{a_2 \leftarrow \{0,1\}^n; r' \leftarrow \{0,1\}^* : (a_2, r')\}$ is the distribution of $\mathcal{S}$'s random tape, we have that the oblivious sampling satisfies the required indistinguishability property.

Finally, to relax to one-way functions we observe that in the OT protocol, the receiver submits two image elements for which it only knows the preimage for only one of them and the other one is obliviously sampled. This can be achieved by using Naor's statistically binding commitment scheme that further has a pseudorandom range.

Our commitment scheme. We next generalize our construction from Section 4.1 and introduce an instanced-dependent commitment scheme based on robust randomized encoding as defined in Definition 2.10. Given an NP language $\mathcal{L}_{\mathcal{R}}$, where $\mathcal{R}$ is the associated relation, and an input statement x, we define the function:

$$f_x(w) = \mathcal{R}(x, w).$$

Namely, the function that checks membership in the NP relation corresponding to the statement x. When it is clear from context we will drop the subscript x and simply refer to it by f. Corresponding to a function f, let $\widehat{f}$ denote the robust randomized encoding that comprises of functions $\widehat{f}_{\text{OFF}}(r)$ (the offline part) and $\widehat{f}_{\text{ON}}(x, r)$ (the online part). We further recall that the simulator's output is captured via two strings $(s_{\text{OFF}}, s_{\text{ON}})$ that correspond to the offline/online parts of the encoding. Our construction is introduced in Figure 5.

[12] More precisely, to generate the random tape for the simulated receiver, we rely on the fact that the commitment scheme (used to implement the OT invocations) has a pseudorandom range.

Instance-Dependent Commitment from Robust Randomized Encoding

Building block: Let com denote a pseudorandom perfectly binding commitment scheme.

Inputs: Let function f be as above and let x denote a statement $x \in \mathcal{L}$.

The commitment scheme:

- $\mathsf{Com}(x, 0)$: S samples r and sends to the receiver $(\widehat{f}_{\text{OFF}}(r), \sigma)$ where $\sigma = \mathsf{com}(r)$.

  Decommitment: S decommits σ to the value r to the receiver, that verifies that the encoding was computed correctly with randomness r and that the decommitment information is correct.

- $\mathsf{Com}(x, 1)$: S computes $(s_{\text{OFF}}, s_{\text{ON}}) \leftarrow \mathcal{S}(1; r')$ and sends (s_{OFF}, σ) to the receiver, where $\sigma \leftarrow \{0,1\}^t$ and $t = |\mathsf{com}(r)|$.

  Decommitment: S sends s_{ON} to the receiver and explains σ as an obliviously generated commitment. The receiver checks if $B(s_{\text{OFF}}, s_{\text{ON}}) = 1$.

- $\mathsf{Com}'(x)$: S generates a commitment as for the case of $\mathsf{Com}(x, 0)$ using randomness r, i.e., it sends $(\widehat{f}_{\text{OFF}}(r), \sigma = \mathsf{com}(r))$.

- $\mathsf{Adapt}(x, \omega, c, 0, r)$: If $\mathsf{Com}'(x; r) \neq c$, then the algorithm returns $\perp$. Otherwise, it outputs the bit 0 and the randomness for computing σ as a commitment to r.

- $\mathsf{Adapt}(x, \omega, c, 1, r)$: If $\mathsf{Com}'(x; r) \neq c$, then the algorithm returns $\perp$. Otherwise, it computes $r' \leftarrow \mathsf{ORE}(f, (x, \omega), r)$ and outputs the bit 1 and r', and further explains σ as an obliviously generated commitment.

Figure 3: Instance-dependent commitment from robust randomized encoding

Theorem 5.2. *Assume the existence of one-way functions. Then, the protocol presented in Figure 5 is a secure adaptive instance-dependent commitment scheme for any language in NP.*

The proof follows analogously to the proof of Theorem 4.1. On a high-level, hiding is achieved by the privacy of the randomized encoding and the hiding of the commitment scheme com. Adaptivity is achieved due to the obliviousness of the randomized encoding and the pseudorandomness of com. Finally, binding follows from the robustness property where in case that $x \notin \mathcal{L}$, then f_x can never output the value 1. Hence, by robustness it follows that no adversary can produce a commitment (z, σ) such that there exists a string r for which $z = \widehat{f}_{\text{OFF}}(r)$ and $\sigma = \mathsf{com}(r)$ (namely, a decommitment to 0), and at the same time produce an online part z' as part of a decommitment to 1 such that $B(z, z') = 1$ since 1 is not in the range of f_x.

5.1.2 Input-delayed Zero-Knowledge Proofs

In this section, we extend the basic construction of instance-dependent commitment schemes from our previous construction to additionally allow constructing input-delayed zero-knowledge proofs. In [LS90], Lapidot and Shamir provided a three-round witness-indistinguishable (WI) proof of knowledge for Graph Hamiltonicity with a special "input-delayed" property: namely, the prover uses the statement to be proved only in the last round. Recently, in [CPS+15] it was shown how to obtain efficient input-delayed variants of the related "Sigma protocols" when used in a restricted setting of an OR-composition.

In this section, we show how randomized-encoding that is secure against adaptive chosen inputs can

be used to realize input-delayed zero-knowledge proofs. Then relying on the recent construction of such a randomized encoding [HJO$^+$15] we obtain a constant-rate input-delayed zero-knowledge proof, namely whose communication complexity is $O(s) + \mathsf{poly}(\kappa)$ where s is the size of the circuit realizing the NP-relation and κ is the security parameter.

We achieve this in two steps. First, we extend our notion of instance-dependent commitment scheme to one where the actual commitment scheme do not require the input statement. Then using such an instance-dependent commitment scheme we will show how to realize an input-delayed zero-knowledge proofs. We provide next definitions for the above primitives.

Our first notion is that of input-delayed instant-dependent commitment scheme. On a high-level, this primitive is a variant of the plain instant-dependent commitment scheme where the real and fake commitment algorithms do not require the knowledge of the input statement in the commit phase. The statement can be adaptively chosen based on the commit phase and will be required only in the decommit phase. Second, we will not require an Adapt algorithm that can explain a fake commitment as an honest commitment of any message by generating random coins for an honest committer that would have produced the same commitment. Instead we will only require the slightly weaker property of the fake commitment being equivocable. Towards this, we will introduce a decommitment algorithm for the honest commitment that additionally takes as input the statement x and produces a decommitment to the corresponding message m. The receiver then verifies the decommitment with respect to the statement x. Corresponding to the fake commitment algorithm, we now require an algorithm that, given the statement and the witness can reveal a commitment (i.e. produce decommitments) to any message m.

Definition 5.3 (Input-delayed IDCS)**.** *Let $\mathcal{R}$ be an NP relation and $\mathcal{L}$ be the language associated with $\mathcal{R}$. A (non-interactive) instance dependent commitment scheme (IDCS) for $\mathcal{L}$ is a tuple of probabilistic polynomial-time algorithms $(\widetilde{\mathsf{Com}}, \widetilde{\mathsf{Com}}', \mathsf{Adapt})$, where:*

- *$\widetilde{\mathsf{Com}}$ is the commitment algorithm: For a message $m \in \mathcal{M}$, and a random string $r \in \{0,1\}^{p(n)}$, $\widetilde{\mathsf{Com}}(1^n, m; r)$ returns a commitment value c where n is the length of the input-instance and $p(\cdot)$ is a polynomial.*
- *$\widetilde{\mathsf{Decom}}$ is the decommitment algorithm that on input a statement x, commitment c, mesage m and randomness r outputs a decommitment d.*
- *$\widetilde{\mathsf{Ver}}$ is the verification algorithm that on input x, m, c, d outputs accept or reject.*
- *$\widetilde{\mathsf{Com}}'$ is a "fake" commitment algorithm: For a random string $r \in \{0,1\}^{q(n)}$, $\widetilde{\mathsf{Com}}'(1^n, r)$ returns a commitment value c where n is the length of the input instance and $q(\cdot)$ is a polynomial.*
- *Equiv is an equivocation algorithm: Let $x \in \mathcal{L}$ and $\omega \in \mathcal{R}_x$. For all c and $r \in \{0,1\}^{q(|x|)}$ such that $\mathsf{Com}'(r) = c$, and for all $m \in \mathcal{M}$, $\mathsf{Equiv}(x, \omega, c, m, r)$ outputs d such that $\widetilde{\mathsf{Ver}}(x, m, c, d)$ outputs accept.*

The hiding property now requires that for any message m, an honest commitment and decommitment to m be indistinguishable from a fake commitment and decommitment to m even when the input statement is adaptively chosen after the commitment phase. The binding property on the other hand will require that for any commitment c and a false statement $x \notin \mathcal{L}$, there exists no values m, d and m', d' such that $\widetilde{\mathsf{Ver}}(x, m, c, d) = \widetilde{\mathsf{Ver}}(x, m', c, d') = \mathsf{accept}$.

We next describe our input-delayed zero-knowledge proof.

Input Delayed Zero-Knowledge Proof for any Language $\mathcal{L} \in \mathsf{NP}$

Building block: Instance-dependent commitment scheme $(\widetilde{\mathsf{Com}}, \widetilde{\mathsf{Com}}', \mathsf{Adapt})$ for language $\mathcal{L}$.

Inputs: A circuit C that computes the function $f'(x, \omega) = (\mathcal{R}(x, \omega), x)$ and a public statement $x \in \mathcal{L}$ for both. A witness ω for the validity of x for the prover $\mathcal{P}$.

The protocol:

1. $\mathcal{P} \rightarrow \mathcal{V}$: $\mathcal{P}$ invokes $\mathsf{com} \leftarrow \widetilde{\mathsf{Com}}'(1^\kappa; r)$ and sends com to the verifier.
2. $\mathcal{V} \rightarrow \mathcal{P}$: The verifier sends a random challenge bit $b \leftarrow \{0, 1\}$.
3. $\mathcal{P} \rightarrow \mathcal{V}$: Upon receiving the input statement x and witness ω,
   - If $b = 0$ then the prover invokes $\mathsf{Equiv}(x, \omega, c, 0, r)$ to obtain a decommitment of c to the bit 0 and sends it to the Verifier.
   - If $b = 0$ then the prover invokes $\mathsf{Equiv}(x, \omega, c, 1, r)$ to obtain a decommitment of c to the bit 1 and sends it to the Verifier.
4. The verifier checks that the decommitment is valid with respect to x.

Figure 4: Input delayed zero-knowledge proof for any language $\mathcal{L} \in \mathsf{NP}$

Theorem 5.4. *Assume the existence of one-way functions. Then, the protocol presented in Figure 4 is an input-delayed honest verifier zero-knowledge proof for any language in NP.*

Proof: Briefly, the security of our proof is implied by the security of the instance-dependent commitment scheme. More concretely, completeness follows easily from the adaptivity property of the commitment scheme. Namely, since an honest prover knows a witness for x, it can always provide a valid reply to any challenge posed by the verifier. Next, we claim that the proof is sound. This is due to the underlying binding property of the commitment scheme. Specifically, for any $x \notin \mathcal{L}$, a corrupted prover cannot convince the verifier in its validity with probability better than $1/2$, as that requires answering both challenges. The binding property of our commitment schemes ensure that the prover cannot do that.

Finally, we discuss zero-knowledge for an honest verifier. Here, the simulator guesses the challenge of the verifier in advance and computes the first message according to this challenge following the honest commitmetn algorithm. In more detail, for the bit b, the simulation relies on the honest commitment algorithm $\widetilde{\mathsf{Com}}(b; r)$ instead of the fake commitment algorithm as the honest prover does. Indistinguishability of the simulation follows directly from the hiding property of the fake commitment algorithm. ■

Finally, we need to show how our input-delayed IDCS can be constructed from a robust randomized encoding that is secure against an adaptive chosen input. On a high-level our instance-dependent commitment scheme from randomized encoding will follow the same approach as in Section 5.1.1. We begin with a randomized encoding for the following function f:

$$f(x, \omega) = (\mathcal{R}(x, \omega), x).$$

Since the randomized encoding is secure against adaptive choice of inputs, we can split the simulation algorithm to an offline and online part, namely $\mathcal{S}_{\text{OFF}}$ and $\mathcal{S}_{\text{ON}}$ where $\mathcal{S}_{\text{OFF}}$ on input 1^n and randomness r' outputs the offline part of the encoding s_{OFF} and $\mathcal{S}_{\text{ON}}$ on input $(1^n, f(x), r')$ outputs the online part s_{ON}.

Instance-Dependent Commitment Scheme from Robust Randomized Encoding secure against Adaptive Choice of Inputs

Building block: Let com denote a pseudorandom perfectly binding commitment scheme.

Inputs: Let function f be as above.

The commitment scheme:

- $\mathsf{Com}(0)$: Sample r and output $(\widehat{f}_{\text{OFF}}(r), \sigma)$ where $\sigma = \mathsf{com}(r)$. Observe that $\widehat{f}_{\text{OFF}}$ does not take the statement x as input.

  $\mathsf{Decom}(x, c, 0, r)$: The decommitment simply contains the decommitment of σ to the value r to the receiver. Ver algorithm outputs accept only if the (offline) encoding was computed correctly with randomness r and that the decommitment information of σ is correct. Otherise Ver outputs reject.

- $\mathsf{Com}(1)$: Compute $s_{\text{OFF}} \leftarrow \mathcal{S}_{\text{OFF}}(1^n; r')$ and output (s_{OFF}, σ), where $\sigma \leftarrow \{0,1\}^t$ and $t = |\mathsf{com}(r)|$.

  $\mathsf{Decom}(x, c, 1, r)$: The decommitment contains $s_{\text{ON}} \leftarrow \mathcal{S}_{\text{ON}}(1^n, (1, x), r')$ and explaination of σ as an obliviously generated commitment, i.e. a random string. Ver algorithm computes $B(s_{\text{OFF}}, s_{\text{ON}})$ and outputs accept only if it evaluates to $(1, x)$. Otherwise it outputs reject

- $\mathsf{Com}'()$: Is identical to $\mathsf{Com}(0)$, i.e. output $(\widehat{f}_{\text{OFF}}(r), \sigma = \mathsf{com}(r))$.

- $\mathsf{Equiv}(x, \omega, c, 0, r)$: If $\mathsf{Com}'(r) \neq c$, then the algorithm returns $\perp$. Otherwise, it outputs the bit 0 and the randomness for computing σ as a commitment to r.

- $\mathsf{Equiv}(x, \omega, c, 1, r)$: If $\mathsf{Com}'(x; r) \neq c$, then the algorithm returns $\perp$. Otherwise, it sends $\widehat{f}_{\text{ON}}((x, \omega), r)$ and further explains σ as an obliviously generated commitment. Recall that the receiver now checks if $B(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}((x, \omega), r)) = (1, x)$.

Figure 5: Instance-dependent commitment from robust randomized encoding

Observe that both the honest and fake commitment algorithms do not depend on the input statement. This is enabled by the adaptive input security of the randomized encoding. The hiding property of the commitment for bit 0 holds directly, whereas the hiding property for the bit 1 follows from the simulation property of the randomized encoding. Binding on the other hand follows directly from the robustness property of the randomized encoding.

The work of Hemenway et al. [HJO$^+$15] shows how to obtain a randomized encoding secure against adaptively chosen inputs. We show in the next section how to extend it to achieve the stronger robustness property. Combining their work with our construction, we have the following corollary.

Corollary 5.5. *Assuming the existence of one-way functions. Then for any NP-relation $\mathcal{R}$, there exists an input-delayed ZK proof with communication complexity $O(s\mathsf{poly}(k))$ where s is the size of the circuit computing the NP relation.*

Additionally, our protocol only depends on an underlying randomized encoding that implements a related functionality in a black-box way.

5.2 Realizations of Our Randomized Encoding

A realization based on garbled circuits. We next demonstrate that our construction can be realized based on garbling schemes. This is shown by proving that Yao's protocol from [LP09] maintains all the security requirements specified above. To this end, the only non-trivial arguments are regarding the following items. First, regarding the proof for which the protocol is secure in the presence of static corruption of P_1 and adaptive (post-execution) corruption of P_2. Second, it must be shown that the adaptive simulation of (post execution corrupted) P_2 can be explained as if P_2 was statically corrupted. The former argument can be easily shown due to the fact that we instantiate the OT calls using a variant of the [EGL85] protocol which allows for receiver equivocation, which is exactly what is required in order to equivocate P_2 internal state. This is because P_2 only uses its input in the OT executions in this protocol.

To prove the later we must modify the way Lindell and Pinkas design their static simulation when P_2 is corrupted. We recall that when P_2 is statically corrupted, the simulator in their proof constructs a fake garbled circuit that always outputs the correct output of P_2. This fake garbling involves a sequence of four ciphertexts per gate that encrypt the same input label four times (the so called the "active" key). On the other hand, in case P_2 is adaptively corrupted, upon statically corrupting P_1, the garbled circuit is honestly generated by the semi-honest corrupted P_1. Now, since it is not possible to explain an honestly generated garbled circuit as a fake one (at least not with the set of tools used for garbling), we slightly modify the original simulation of Lindell and Pinkas as follows. Instead of having four ciphertexts that encrypt the same plaintext, the simulator generates only one valid ciphertext and three oblivious ciphertexts. Note that now it is possible to explain an adaptive simulation of P_2 as a static one by relying on the obliviousness property of the underlying encryption scheme.

A realization based on [GMW87]. We consider the basic protocol from [GMW87] where parties P_1 and P_2 first XOR-share their respective inputs x and y into two shares (x_1, x_2) and (y_1, y_2), and exchange one share with each other, say x_2 from P_1 and y_1 from P_2. Next, they evaluate the circuit gate by gate where given the shares of the input they try to obtain shares for the output. The shares that correspond to the output of an addition gate can be simply obtained by locally adding that shares that correspond to the input. Multiplication gates, on the other hand, require oblivious transfer. For instance, if a_i, b_i are the input shares held by party P_i ($i \in \{1, 2\}$) where the inputs are $a_1 \oplus a_2$ and $b_1 \oplus b_2$, then to compute the product the parties engage in a 1-out-of-4 OT where P_2 sets its input as (a_2, b_2) and P_1 sets its inputs as $\{(a_1 \oplus A)(b_1 \oplus B) + s\}_{A\in\{0,1\},B\in\{0,1\}}$ where s is chosen at random. In essence, corresponding to the input (a_2, b_2), P_2 learns $(a_1 \oplus a_2)(b_1 \oplus b_2) + s$ and uses that as its output shares, while P_1 uses s. Finally, P_1 transmits its share of the output wires to P_2. In this protocol P_1 is the designated sender for all OT invocations and the protocol admits (UC) simulation in the presence of adaptive adversaries corrupting either P_1 or P_2 in the OT-hybrid.

We next discuss how to construct a robust randomized encoding with oblivious sampling from the [GMW87] protocol, that satisfies all the required properties. Namely, achieving privacy against semi-honest adversaries with UC security and requiring that P_1 is the designated sender are immediately satisfied by the protocol (in the OT-hybrid). Robustness, on the other hand, holds because the protocol achieves unconditional security in the OT-hybrid in the information theoretic setting against passive adversaries.

Finally, we argue oblivious sampling. Recall that the simulation for party P_2 in the [GMW87] protocol proceeds as follows. Given the input y and output $f(x, y)$ of the computation, the simulation generates first the shares of P_2's input, namely (y_1, y_2), and a random value x_2 for the share of P_1's input that it transmits. Next, for every OT call, it simply sets the output of the OT invocation as a random bit. It then follows P_2's computation to generate a share z_2 of the output. Finally, it sets the share of the output transmitted by P_1 as

$z_1 = f(x, y) \oplus z_2$. To demonstrate oblivious sampling, we need to show that any view of P_2 can be shown as an output of the simulation. Recall that the view of P_2 includes its input and output, random tape and the messages from P_1. The messages from P_1 are x_2 (share of P_1's input), the OT outputs corresponding to P_2's queries which are all uniformly distributed because they are masked by P_1, and the share of the output. Given the view, the message x_2 and all outputs of the OT invocations can be placed in the random tape of the simulation as they are randomly generated. The final message z_1 is fixed and determined by x_2, the OT outputs and the random tape of P_2. Therefore, the protocol satisfies oblivious sampling in the OT-hybrid.

Realizing robust randomized encoding secure against adaptive choice of inputs based on [HJO$^+$15]. On a high-level, the construction [HJO$^+$15] considers the randomized encoding based on garbled circuits and then modifies it to achieve security against adaptive chosen inputs using the following approach: The offline encoding now includes an encryption of the garbled circuit under a special kind of encryption scheme. This encryption scheme allows the encryptor to reveal a ciphertext to M different possible plaintexts (with certain restrictions) where M is a parameter chosen for the security proof.[13] In fact, given the parameter M and the encryption key the encryptor generates M distinct keys corresponding to the plaintexts. We note that the honest encoding and the simulated encoding will not use this equivocation property, and reveal only one key corresponding to the intended plain text. However, the proof of security involves a sequence of hybrid steps that will make use of the different keys.

Recall that a robust randomized encoding $f = \widehat{f}_{\text{OFF}}, \widehat{f}_{\text{ON}}$ requires that there does not exist any random tape r for the encoding such that given an encoding generated using randomness r, namely $\sigma_{\text{OFF}} = \widehat{f}_{\text{OFF}}(r)$, there is no string σ_{ON} such that the decoder B on input $(\sigma_{\text{OFF}}, \sigma_{\text{ON}})$ outputs an element outside the range of f. As such, the construction presented in the work [HJO$^+$15] is not necessarily robust as an encoding, even if honestly generated, can be revealed to M different possible plaintexts, therefore possibly M different garbled circuits. We modify the construction to make it robust by including in the offline encoding a non-interactive statistically binding commitment of the key. More precisely, we will require that $\widehat{f}_{\text{OFF}}$ also include M commitments to the encryption key used to encrypt the garbled circuit. The online encoding, now, along with the key and the online encoding corresponding to the garbled circuit, includes the decommitment of a randomly chosen commitment among the M commitments. The decoder verifies that the key revealed is indeed the one decommitted, decrypts the ciphertext and evaluates the garbled circuit. This scheme is robust as any randomness revealed for an offline encoding will have to show that all the M commitments are commitments to the same key. Since the commitments are perfectly binding, this means it is impossible for a correctly generated offline phase to be revealed to more than one garbled circuit via the onling encoding. Now, since there is only one garbled circuit that can be revealed, we can conclude the robustness of the scheme using the robustness of the garbled circuit construction. However, we still require that the proof of security from [HJO$^+$15] holds. This proof of security requires that the real and simulated encoding are indisitinguishable. Since in this indistinguishability experiment, the randomness of the offline phase is not revealed and only one commitment is decommitted to in the online encoding, it is possible to, in hybrid steps, commit to M different keys and reveal the one corresponding to the one required in the hybrid step.

The efficiency of our randomized encoding. As shown above, both garbled schemes [Yao86, LP09] and the [GMW87] protocol satisfy the required properties to realize our randomized encoding. Thus, if we rely on the former protocol, the offline complexity is $O(s\mathsf{poly}(\kappa))$ whereas the online complexity is $n\mathsf{poly}(\kappa))$, where s is the size of the circuit computing f, n is the input length of f and κ is the security parameter. In contrast, if we rely on [GMW87], we get that the online and offline complexities are both $O(s\mathsf{poly}(\kappa))$.

[13] M will be chosen to be proportional to the width of the circuit implementing the function f.

Finally, our robust randomized encoding secure against adaptive input based on [HJO+15] has an offline efficiency of $O(s\mathsf{poly}(\kappa))$ and an online efficiency of $O((d+n)\mathsf{poly}(\kappa))$ where d is width of the circuit implementing the function under consideration.

6 Instance-Dependent Commitments for a Bounded Message Space

In this section we present a more efficient instance-dependent commitment scheme with a better equivocation parameter (for message domains larger than $\{0,1\}$). Formally, we say that an instance-dependent commitment has equivocation parameter ℓ if the message space of the commitment scheme is $\{0,1\}^\ell$. In particular, given an arbitrary integer ℓ we show how to construct a commitment scheme with an equivocation parameter ℓ. Let $\mathcal{L}_\mathcal{R}$ be the NP language associated with relation $\mathcal{R}$ and let C be the circuit that on input (x,ω) outputs 1 iff $(x,\omega)\in\mathcal{R}$. Then improving the binary equivocation parameter from Section 4.1 using garbled schemes requires taking a new approach as it is not a matter of generating a standard garbled circuit or a simulated one any longer.

Our idea is to use the "IPS-technique" from [IPS08] which we briefly recall for the two-party setting. In the IPS compiler we start with a multiparty protocol among n parties that is information-theoretically secure when a majority of the parties are honest. This is referred to as the *outer protocol*. This outer protocol is simulated by the actual parties P_1 and P_2 via a two-party protocol secure against semi-honest adversaries which is referred to as the *inner protocol*. The parties in the outer protocol are called the *servers* and the actual parties P_1 and P_2 are called the *clients*. The high-level approach is to make P_1 and P_2 engage in n sub-protocols $\rho_1,\ldots,\rho_n$ where in ρ_j, the parties jointly compute the next message of server S_j. In typical instantiations of this compiler the (simulated) servers themselves do not have any input. The clients P_1 and P_2 share their inputs with the n servers via a verifiable secret sharing scheme, for which the servers then use to securely compute an appropriate function over their joint inputs. Then, for every step in the computation of server S_j, P_1 and P_2 securely execute the next-message functionality for the outer protocol using ρ_j, to produce the output (i.e., the next message of S_j) which is secret shared between the clients.

Note that if the inner protocols are secure against malicious parties then the entire protocol is maliciously secure as well. However, the requirements on the inner protocol is mild, namely, it requires only privacy against semi-honest adversaries. Therefore, there needs to be a mechanism for the parties to enforce (check) honest behavior of the other party. To handle this issue, a novel concept called *watchlists* was introduced by [IPS08]. In essence, each party gets to check the other party's behavior on a subset of the servers that are on its watchlists. These subsets are randomly selected by both parties for which the corresponding party has complete information of the servers in the selected subset, namely, their inputs, incoming messages from other servers and output. With this information, each party can verify that the other party performed the computation correctly for the servers on its watchlists. The number of servers in the watchlists should be carefully chosen as it should not be too high to avoid compromising the privacy of the outer protocol and not be too low to allow catching misbehavior of each party with sufficiently high probability. It was shown in [IPS08] that in the two-party setting $n=O(\kappa)$ servers is sufficient as long as $t=O(n)$ is the watchlists size. For more optimizations, we refer the reader to [LOP11]. We now proceed to give more details regarding the inner protocol and watchlist mechanism to the extent that is relevant to our work. For more details and discussion we refer the reader to [IPS08, LOP11].

To understand what computation is performed in the inner protocol we need to observe the requirements on the outer protocol. First, the servers do not start with any input and receive their inputs from the clients and therefore their actions are a function of their random tapes and the incoming messages. Next, if the outer protocol is interactive then the functionality executed by the inner protocol needs to be *reactive*. Namely, in

each round of the outer protocol, each server starts with a state computed from a previous round and the set of messages received from other servers in the previous round, and outputs a state and messages to be sent to other servers. The input for each round is secret shared among P_1 and P_2. The inner protocol outputs the sharing of the state and the messages sent by each server to the other party. In order for the inner protocol to work, parties P_1 and P_2 must accomplish two things regarding each server S_j:

1. The servers' computation must be performed according to some random tape.

2. The parties must correctly carry the share of the state of S_j and the shares of messages received by S_j from the other servers from one round to the next one, where the later messages are, in turn, the outputs from the computations performed for the other servers in the previous round.

These two conditions are exactly what each party needs to enforce with respect to the other party in order to prevent cheating, which is carried out using a watchlist mechanism that works as follows. Suppose P_2 wants to check the behavior of party P_1. Then P_1 chooses random tapes $r_1^1, \ldots, r_1^n$ where r_1^j is the random tape that it uses for ρ_j. It also selects keys $k_1^1, \ldots, k_1^n$ to a semantically secure encryption scheme where the share of the messages from any server to S_j, computed by P_1, is encrypted using k_1^j and explicitly passed to P_2. In essence, if P_2 knows r_1^j and k_1^j it can recompute the messages that P_1 should have sent in the execution using ρ_j. We remark here that there are two inputs corresponding to the server in each round that P_1 uses, the share of the incoming messages and the share of the state. While the incoming messages from the other servers can be obtained by P_2 by decrypting the encryptions sent by P_1 in the previous round using key k_1^j, the share of the state in each round is internally maintained by P_1. Nevertheless, using r_1^j and k_1^j, P_2 can obtain the share of the state that should have been computed in each round to verify the actions of P_1. Finally, P_2 obtains r_1^j, k_1^j corresponding t servers by running a t-out-of-n oblivious transfer where P_1 acts as the sender and sets its input as $((r_1^1, k_1^1), \ldots, (r_1^n, k_1^n))$ and P_2 uses the indices of the servers that it wants to put on its watchlist as its input. P_1 can analogously check P_2.

In our construction, we only require P_2 to check the actions of P_1, since in our instance-dependent commitment scheme the receiver always receives the complete information regarding P_1 and can itself verify the semi-honest behavior (of all servers). We now proceed to describe our instance-dependent commitment scheme, starting with the requirements from our outer and inner protocols.

Outer protocol. We consider an *outer protocol* Π that is executed among two clients, denoted by C_1, C_2, and a set of $n = O(\kappa)$ servers $\{S_j\}_{j\in[n]}$, computing the following functionality:

$$g(\omega_1, \omega_2) = \mathcal{R}(x, \omega_1 \oplus \omega_2)$$

where ω_i is client C_i's input and the servers do not have any inputs. We require that Π UC realizes the functionality f, against adaptive corruptions of up to t servers, adaptive corruption of C_2 and static corruption of C_1. Additionally, just as in [IKOS09, IPS08] we need the outer protocol to be t-robust against malicious adversaries (see Definition 2.15). The corruptions are active and the security is computational. Concretely, we instantiate the outer protocol with the protocol of Ben-Or, Goldwasser and Micali [BGW88, AL11]. This description slightly differs from the preceding discussion, where the outer protocol just involves n servers, as we have two additional clients involved in the outer protocol. The only purpose of the clients is to distribute the inputs to the servers and finally collect the result of the computation. The actions of the clients can (and will) be performed by P_1 and P_2.

Inner protocol. The inner protocol is executed between the parties P_1 and P_2 that simulate the outer protocol. More precisely, there are n executions of the inner protocol $\rho_1, \ldots, \rho_n$, where protocol ρ_j computes the actions of S_j. ρ_j is required to be secure against (passive) semi-honest adversaries tolerating a static corruption of P_1 and an adaptive corruption of P_2 in the OT-hybrid. We instantiate the inner protocol with the [GMW87] semi-honest two-party protocol in the OT-hybrid where as in the previous section, we implement the OT invocations using one-way permutations as specified. We can then relax this assumption to one-way functions using an approach that is similar to the one used in the previous section. As mentioned above, we require from P_2 to monitor the behavior of P_1 using watchlists. This is implemented by making P_2 sample indices $s_1, \ldots, s_t$ where each s_i is sampled uniformly at random from $[n]$. Then before the execution begins, the parties engage in an OT protocol where P_1 acts as the sender and sets its input as $((r_1^1, k_1^1), \ldots, (r_1^n, k_1^n))$ and P_2 sets its input as $s_1, \ldots, s_t$, where r_1^j is the randomness that P_1 is supposed to use in the simulation of server S_j and k_1^j is the key used by P_1 to encrypt the shares of all incoming messages computed in the inner protocol from any server to S_j, for which this ciphertext is sent to P_2.

Let Π_g^{OT} be the combined protocol and let $\widetilde{\Pi}$ be the protocol that relies on one-way functions to realize the OT calls as in the previous section. We also consider a slightly modified algorithm for P_2 that in addition to ω_2 takes as input the indices $\mathbf{s} = (s_1, \ldots, s_t)$ of the t servers that are to be on its watchlist. We require the following additional properties to be satisfied by our protocol:

Semi-adaptive security. We require privacy against a (semi-honest) adversary $\mathcal{A}_1$ that statically corrupts P_1 and corrupts P_2 post execution. In other words, we require a simulation that first generates the view of P_1 given its input and output. Next, given the input and output of P_2 it can provide a view of P_2 consistent with the view of P_1 (namely the transcript of messages received to and from P_1 must be consistent with the same messages in the view of P_1). This property follows essentially using the same proof as in [IPS09] which demonstrates that the combined protocol satisfies full adaptive security in OT-hybrid if the inner protocol satisfies the same. Here, we only require semi-adaptive security and we can relax the OT requirement to the same semi-adaptive security which is satisfied in our instantiation. Nevertheless, for completeness, we discuss in Section 6.2 why our combined protocol satisfies this semi-adaptive security.

Oblivious sampling. We recall the oblivious sampling property from our previous section that we need our inner protocol to satisfy. Let $\mathcal{S}_2$ be the simulation corresponding to an adversary that statically corrupts P_2. We require an algorithm $\mathsf{InvSamp}$ that on input a view of P_2, $(\tau, ((\omega_2, \mathbf{s}), r_2))$ outputs random coins r' such that $\mathcal{S}_2$ on input $(\omega_2, f(\omega_1, \omega_2) = 1)$ and random tape r' outputs the view $(\tau, ((\omega_2, \mathbf{s}), r_2))$ where τ is the transcript of messages exchanged and $(\omega_2, \mathbf{s})$ and r_2 are the input and random tape of P_2. Furthermore, we require that $\mathsf{InvSamp}$ on input a view generated by $\mathcal{S}_2$ with random tape r' returns the same random tape r'. As discussed in Section 5.2 this property is met by the [GMW87] protocol.

Unique randomness. We require that given the transcript τ of interaction between P_1 and P_2 using $\widetilde{\Pi}$ and the input ω_2 of P_2 then there exists at most one view for each S_j that is consistent with τ and P_2's input ω_2. Towards achieving this, we require that given a transcript of the interaction τ and an input $(\omega_2, \mathbf{s})$ of P_2, there exists at most one value for the random tape used by P_2 that is consistent with the transcript. This is because, given τ, it holds that P_1's view is completely fixed in the transcript τ as its input and randomness are fixed as part of the watchlist phase. Then the unique randomness property fixes P_2's view on input $(\omega_2, \mathbf{s})$. This, in turn, implies that the views of all emulated servers are determined. Furthermore, the views of the servers determined this way are independent of $\mathbf{s}$ since $\mathbf{s}$ is only used in the watchlist phase and does not affect the P_1's computations in the inner protocol.

We next argue the unique randomness property for our protocol. Recall first that in the [GMW87] protocol in the OT-hybrid, the only randomness required by P_2 is for the input sharing which given the share sent from P_2 to P_1 (that is present in the transcript) and P_2's input results in a fixed random tape for P_2. We next claim that the same property extends to the modified protocol when the OT invocations are instantiated with a one-way permutation. This is because the receiver's inputs and outputs to each OT invocation have already been determined and when instantiated with the one-way permutation, the transcript fixes the image elements used in the OT invocation which, in turn, fixes the random tape of P_2 required for that OT instance.

Next, we return to our combined protocol Π_g^{OT} where the parties interact in several instances of the inner protocol that securely computes a reactive functionality. To show that the combined protocol satisfies the same unique randomness property, we observe that given the transcript of the interaction τ and P_2's input, the input of P_2 in every instance of the inner protocol is determined. Hence, from the unique randomness property of the [GMW87] protocol, it follows that the random tape of P_2 is fixed for each instance of the inner protocol. The only other randomness required by P_2 in the combined protocol is to obtain the watchlist information corresponding to the servers it maintains. Now, since P_2 receives this information via an OT, the same argument as in the previous paragraph ensures that the random tape of P_2 is fixed given the transcript and $\mathbf{s}$. Therefore, we can conclude that our combined protocol satisfies the unique randomness property.

6.1 Our Commitment Scheme

We are now ready to describe our instance-dependent commitment scheme. The building-blocks that we rely on in the construction are (1) a statistically-binding commitment scheme com with pseudorandom range, where such commitments schemes can be based on one-way function [Nao91]. (2) An interactive hashing protocol $(\mathrm{S}_{\mathrm{IH}}, \mathrm{R}_{\mathrm{IH}})$ described in Section 2.3 with parameters $n_1 = t \log n$, $m_1 = t \log n - ct/2$ and a security parameter $\kappa_{\mathrm{IH}} = ct/4$ where c is a constant that is determined later. We construct a commitment scheme whose message space is in $\{0,1\}^{ct/4}$. Hence, to achieve an equivocation parameter ℓ we simply set $t = 4\ell/c$.

Commitment algorithm $\mathsf{Com}(x, m)$**.** Consider the (semi-honest) adversary $\mathcal{A}_2$ that statically corrupts P_2, namely, at the beginning of an execution. At the end of the execution $\mathcal{A}_2$ sends $(\tau, ((\omega_2, \mathbf{s}), r_2))$ to $\mathcal{Z}$ where τ is the transcript of messages exchanged between P_1 and P_2 and $(\omega_2, \mathbf{s})$ and r_2 are the respective input and randomness used by party P_2. Let $\mathcal{S}_2$ be the corresponding simulator. The Com algorithm on input statement x and message $m \in \{0,1\}^{ct/4}$ proceeds as follows:

1. Com samples ω_2 from $\{0,1\}^{|\omega|}$ and $\mathbf{s} = (s_1, \ldots, s_t)$ from $[n]^t$ at random and emulates an execution using $\mathcal{S}_2$ where the input of P_2 is set to $(\omega_2, \mathbf{s})$. $\mathcal{S}_2$ produces at the end a view $(\tau, ((\omega_2, \mathbf{s}), r_2))$ of P_2 where τ is the transcript of messages exchanged between P_1 and P_2. It then sends τ to the receiver.

2. The committer and the receiver engage in the interactive protocol $(\mathrm{S}_{\mathrm{IH}}, \mathrm{R}_{\mathrm{IH}})$ where the committer plays the role of S_{IH} on input $\mathbf{s}$ and the receiver plays the role R_{IH}. Here, the receiver sends h_i in round $i \in [r + ct/4]$ where $h_i \leftarrow 0^{i-1}1a_i$ and a_i is chosen at random from $\{0,1\}^{t \log n - i}$ where $r = m_1 + \kappa_{\mathrm{IH}} = t \log n - ct/2 + ct/4 = t \log n - ct/4$. The committer replies with the message z_i in round i where $z_i = \langle h_i, \mathbf{s} \rangle$ for $i \leq r$ and $z_i = \langle h_i, \mathbf{s} \rangle \oplus m_i$ for $r + 1 \leq i \leq r + ct/4$. Note that $r + ct/4 = t \log n - ct/4 + ct/4 = t \log n$.

We describe the entire transcript by $(\tau, \mathbf{h}, \mathbf{z})$. We remark that the random tape $\tilde{r}$ of Com can be written as $(\omega_2, \mathbf{s}, r', \mathbf{h})$ where $(\omega_2, \mathbf{s})$ is used as P_2's input, r' is the random tape used to execute $\mathcal{S}_1$ and $\mathbf{h}$ is for the interactive hashing used to incorporate m. Then the decommitment to m is the random coins $\tilde{r} = (\omega_2, \mathbf{s}, r', \mathbf{h})$ of Com used to generate the commitment. Using r', the receiver reconstructs the output of $\mathcal{S}_2$ from which it obtains the random tape r_2 of P_2 corresponding to input ω_2. It then verifies that the simulated view of P_2 is consistent with the transcript τ from the commitment phase and that P_2 outputs 1 at the end of the execution. Finally, the receiver accepts if $\langle h_i, \mathbf{s} \rangle = z_i$ for all $i \leq r$ and $\langle h_i, \mathbf{s} \rangle \oplus m_{i-r} = z_i$ for all $r + 1 < i \leq t \log n$.

Fake commitment algorithm $\mathsf{Com}'(x)$**.** Consider a (semi-honest) adversary $\mathcal{A}_1$ that corrupts P_1 at the beginning of the execution. At the end of the execution, $\mathcal{A}_1$ first sends τ to $\mathcal{Z}$ where τ is the transcript of messages exchanged between P_1 and P_2. Next it (adaptively) corrupts P_2 and sends $((\omega_2, \mathbf{s}), r_2)$ to $\mathcal{Z}$ where $(\omega_2, \mathbf{s})$ and r_2 are the respectively input and randomness used by party P_2. Let $\mathcal{S}_1$ be the corresponding simulator as guaranteed by the properties of $\widetilde{\Pi}$. Then the fake commitment algorithm Com' proceeds as follows: It runs $\mathcal{S}_1$ until it sends the first message to the environment. Recall that $\mathcal{S}_1$ statically corrupts P_1 and upon completion of the execution sends the transcript of the messages to the environment. Specifically, Com' generates a transcript τ of the interaction by running $\mathcal{S}_1$ where P_1's input is set to ω_1 chosen at random. It then outputs $(\tau, \mathbf{h}, \mathbf{z})$ where $\mathbf{h} = (h_1, \ldots, h_{t \log n})$ and $\mathbf{z} = (z_1, \ldots, z_{t \log n})$ are chosen at random from the appropriate domains.

Adapt **algorithm.** Recall that the Adapt algorithm needs to demonstrate that the transcript of the commitment phase was generated using the honest algorithm Com and message m, upon receiving as input the witness ω and the message m. Towards this, Adapt first computes $\omega_2 = \omega \oplus \omega_1$ where ω_1 was the input set for P_1 by Com'. Next, it determines the indices $\mathbf{s} = (s_1, \ldots, s_t)$ as follows:

- Using the message m (for which the committer's internal state needs to be explained with respect to), Adapt computes $z^*_{r+i} = z_{r+i} \oplus m_i$ for $1 \leq i \leq ct/4$. Denote the matrix $\mathbf{A}$ where row i contains the bits of h_i. Next define the vector $\mathbf{z}^* = (z_1, \ldots, z_r, z^*_{r+1}, \ldots, z^*_{t \log n})$. We solve the set of linear equations $\mathbf{As} = \mathbf{z}^*$ to obtain $\mathbf{s}$. The set of equations are always solvable since from the way $h_1, \ldots, h_r$ are chosen by the interactive hashing protocol and the way $h_{r+1}, \ldots, h_{t \log n}$ are chosen, we have that $\mathbf{h} = (h_1, \ldots, h_{t \log n})$ contains $t \log n$ linearly independent vectors.

Adapt, uses $\mathcal{S}_1$ to generate the view of P_2 relative to τ (the transcript generated by Com'), by adaptively corrupting P_2 post execution, where P_2's input is set to $(\omega_2, \mathbf{s})$. Recall that such a simulation exists following the semi-adaptive simulation property of our combined protocol. Finally, using a view of P_2, Adapt uses the invertible sampler algorithm $\mathsf{InvSamp}$ to demonstrate that it was generated using the simulator $\mathcal{S}_2$. More precisely, it runs $\mathsf{InvSamp}$ on τ to generate r' and outputs $r^* = (\omega_2, \mathbf{s}, r', \mathbf{h})$.

Complexity. For any integer t, the commitment phase incurs a cost proportional to the communication complexity of our combined protocol in the OT-hybrid and the communication complexity of the incorporated OT calls. If we rely on the outer and inner protocols as specified at the beginning of this section then the communication complexity of our combined protocol and the number of OT invocations are both bounded by $O(s) + t \cdot \mathsf{poly}(\kappa, d, \log s)$, where s is size of the circuit, d is the depth of the circuit and κ is the security parameter. Therefore, the overall communication complexity is $O(s) \cdot \kappa + t \cdot \mathsf{poly}(\kappa, d, \log s)$. If we rely on one-way functions, then the overhead of each OT invocation is $\mathsf{poly}(\kappa)$. Therefore, for any equivocation

parameter ℓ, we set $t = O(\ell)$ implying communication complexity $O(s) \cdot \mathsf{poly}(\kappa) + \ell \cdot (\kappa, d, \log s)$. In comparison, both our previous constructions, the one based on garbled circuits from Section 4.1 and the more general construction based on robust randomized encodings from Section 5.1.1, result in $O(s\ell \cdot \mathsf{poly}(\kappa))$ overhead where the communication incurs a multiplicative cost of ℓ. In contrast, our more efficient scheme from this section incurs an additive cost of $\widetilde{O}(\ell)$. We further note that the round complexity of our protocol is $O(t \log n)$ which is incurred by the round complexity of the interactive hashing.

We next prove the following theorem,

Theorem 6.1. *Assume the existence of one-way functions. Then, the above protocol is a secure adaptive instance-dependent commitment scheme for any language in NP with communication complexity $O(s) \cdot \mathsf{poly}(\kappa) + \ell \cdot \mathsf{poly}(\kappa, d, \log s)$, where ℓ is the equivocation parameter, s is the size of the circuit computing the NP relation, d is the depth of the circuit and κ is the security parameter.*

Proof: The proof follows by demonstrating the three properties from Definition 2.5. As in our previous section, we consider a specific environment $\mathcal{Z}^*$ that assigns inputs to the parties as follows. $\mathcal{Z}^*$ gives P_1 and P_2 inputs ω_1 and ω_2 where ω_1 is chosen at random and $\omega_2 = \omega_1 \oplus \omega$. At the end of the execution $\mathcal{Z}^*$ outputs all the messages received from $\mathcal{A}$ as its output.

Adaptivity and computational hiding. To prove hiding, we need to show that for any malicious receiver R^* the views are indistinguishable when receiving a commitment to any two arbitrary messages m and m'. We prove a stronger result that shows that an honestly generated commitment to a message m using Com along with the decommitment information is indistinguishable from a fake commitment Com' along with a decommitment made to message m using the Adapt algorithm. This implies hiding using a standard hybrid argument, as the fake commitment algorithm does not get the message as an input.

In more detail, we prove that the following two distributions

- $D_1 = \{\mathsf{Com}(x, m; U_{p(|x|)}), m, U_{p(|x|)}\}_{x \in \mathcal{L}, \omega \in \mathcal{R}_{\mathcal{L}}}$, and
- $D_2 = \{\mathsf{Com}'(x; U_{p(|x|)}), \mathsf{Adapt}(x, \omega, \mathsf{Com}'(x; U_{p(|x|)}), m, U_{p(|x|)})\}_{x \in \mathcal{L}, \omega \in \mathcal{R}_{\mathcal{L}}}$

are computationally indistinguishable.

Recall the difference between Com' and Com is in that Com' generates the internal transcript of protocol $\widetilde{\Pi}$ using the simulation of $\mathcal{A}_1$ while Com generates it using the simulation of $\mathcal{A}_2$. Moreover, the transcript of the commitment phase contains the transcript of the messages between P_1 and P_2, as well as the tuple $(\mathbf{h}, \mathbf{z})$ where $\mathbf{h} = (h_1, \ldots, h_{t \log n})$ and $\mathbf{z} = (z_1, \ldots, z_{t \log n})$. Intuitively speaking, the latter tuple information theoretically hides the message m as the number of possible values for $(s_1, \ldots, s_t)$ is $[n]^t$. We next formally prove the stronger statement that includes the decommitment information.

- **Hybrid** H_0. This experiment is identically distributed to D_1. From the description of our Com scheme it follows that we can rewrite the distribution (and therefore the ouput of the this experiment) as follows:

  1. Pick $\mathbf{h}$ and $\mathbf{s}$ uniformly at random from their respective domains. Compute $z_i = \langle h_i, \mathbf{s} \rangle$ for $i \leq r$ and $z_i = \langle h_i, \mathbf{s} \rangle \oplus m_{i-r}$ for $r + 1 < i \leq t \log n$.
  2. Pick $\omega_2 \leftarrow \{0,1\}^{|\omega|}, r' \leftarrow \{0,1\}^*$. Set $r_{\mathcal{Z}} = (\omega_2, \mathbf{s})$ and $r_{\mathcal{S}_2} = r'$. Compute

$$(\tau, \mathsf{view}_{P_2}) \leftarrow \mathbf{IDEAL}_{f, \mathcal{S}_2, \mathcal{Z}^*}(\kappa, \omega, (r_{\mathcal{Z}}, r_{\mathcal{S}_2})).$$

We recall here that $\mathcal{Z}^*$ is an environment that on input ω XOR-shares it to ω_1 and ω_2 and provides them as input to P_1 and P_2. Without loss of generality, we assume here that $\mathcal{Z}^*$ uses a portion of its random tape as the share ω_2 and ω_1 is obtained by setting $\omega_1 = w \oplus \omega_2$. Set the commitment phase transcript $\sigma = (\tau, \mathbf{h}, \mathbf{z})$.

3. Output $\big(m, \sigma, (\omega_2, \mathbf{s}, r', \mathbf{h})\big)$.

We remark that view_{P_2} contains $(\omega_2, \mathbf{s})$ and r_2, which are the input and random tape of P_2, as well as the messages sent from P_1 to P_2, whereas τ is the entire transcript of messages between P_1 and P_2.

- **Hybrid** H_1: In this execution, we consider the honestly generated commitment to message m followed by the decommitment that is generated differently. The decommitment contains $(\omega_2, \mathbf{s}, r', \mathbf{h})$ where r' is the random tape used to execute $\mathcal{S}_2$. Instead of using the actual randomness used by $\mathcal{S}_2$ to decommit, we rely on the $\mathsf{InvSamp}$ algorithm. More precisely, given τ from the commitment phase and random coins r, we complete the simulation by $\mathcal{S}_2$ that produces input $(\omega_2, \mathbf{s})$ and randomness r_2 for P_2 and run $\mathsf{InvSamp}$ on input $(\tau, ((\omega_2, \mathbf{s}), r_2))$ to obtain r^*. The decommitment is then set as $(\omega_2, \mathbf{s}, \mathbf{h}, r^*)$. More formally, the output of this experiment is:

  1. Pick $\mathbf{h}$ and $\mathbf{s}$ uniformly at random from their respective domains. Compute $z_i = \langle h_i, \mathbf{s} \rangle$ for $i \leq r$ and $z_i = \langle h_i, \mathbf{s} \rangle \oplus m_{i-r}$ for $r + 1 < i \leq t \log n$.
  2. Pick $\omega_2 \leftarrow \{0,1\}^{|\omega|}, r' \leftarrow \{0,1\}^*$. Set $r_{\mathcal{Z}} = (\omega_2, \mathbf{s})$ and $r_{\mathcal{S}_2} = r'$. Compute

  $$(\tau, \mathsf{view}_{P_2}) \leftarrow \mathbf{IDEAL}_{f, \mathcal{S}_2, \mathcal{Z}^*}(\kappa, \omega, (r_{\mathcal{Z}}, r_{\mathcal{S}_2})).$$

  Set the commitment phase transcript $\sigma = (\tau, \mathbf{h}, \mathbf{z})$.
  3. Output $\big(m, \sigma, (\omega_2, \mathbf{s}, r^*, \mathbf{h})\big)$ where $r^* = \mathsf{InvSamp}(\tau, \mathsf{view}_{P_2})$.

  The only difference between the outputs of H_1 from the previous hybrid is in Step 3 where $\mathsf{InvSamp}$ algorithm. However, from the requirements of our $\mathsf{InvSamp}$ algorithm, it follows $r^* = r'$. Therefore, the output of H_0 and H_1 are identically distributed.

- **Hybrid** H_2: Recall that Com generates the transcript in the commitment phase τ using the simulator $\mathcal{S}_2$ for the underlying protocol. In this hybrid experiment we generate the transcript in the commitment phase differently. We use the witness ω and emulate the adversary $\mathcal{A}_2$ in the real experiment with environment $\mathcal{Z}^*$. We modify the decommitment appropriately. Specifically, in H_1 the random coins for Com are generated by running $\mathsf{InvSamp}$ on the view of P_2 obtained from the output of $\mathcal{S}_2$. On the other hand, in this experiment we run $\mathsf{InvSamp}$ on the view of P_2 output by $\mathcal{A}_2$ in the real world experiment. More formally, the output of this experiment is:

  1. Pick $\mathbf{h}$ and $\mathbf{s}$ uniformly at random from their respective domains. Compute $z_i = \langle h_i, \mathbf{s} \rangle$ for $i \leq r$ and $z_i = \langle h_i, \mathbf{s} \rangle \oplus m_{i-r}$ for $r + 1 < i \leq t \log n$.
  2. Pick $\omega_2 \leftarrow \{0,1\}^{|\omega|}$. Set $r_{\mathcal{Z}} = (\omega_2, \mathbf{s})$ and $r_{\mathcal{A}_2} = \lambda$ (the empty string). Compute

  $$(\tau, \mathsf{view}_{P_2}) \leftarrow \mathbf{REAL}_{\widetilde{\Pi}, \mathcal{A}_2, \mathcal{Z}^*}(\kappa, \omega, (r_{\mathcal{Z}}, r_{\mathcal{A}_2})).$$

  Set the commitment phase transcript $\sigma = (\tau, \mathbf{h}, \mathbf{z})$.
  3. Output $\big(m, \sigma, (\omega_2, \mathbf{s}, r^*, \mathbf{h})\big)$ where $r^* = \mathsf{InvSamp}(\tau, \mathsf{view}_{P_2})$.

The only difference between the experiments H_1 and H_2 is in Step 2 and indistinguishability follows directly from the indistinguishability of the simulation by $\mathcal{S}_2$ since

$$\mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_2,\mathcal{Z}^*}(\kappa,\omega) \stackrel{\mathrm{c}}{\approx} \mathbf{IDEAL}_{f,\mathcal{S}_2,\mathcal{Z}^*}(\kappa,\omega).$$

- **Hybrid** H_3: In this experiment we use the witness ω and emulate the adversary $\mathcal{A}_1$ instead of $\mathcal{A}_2$. The rest of the computation follows exactly as in the previous hybrid. The decommitment is obtained as in the previous hybrid with the exception that the view of P_2 used to run $\mathsf{InvSamp}$ is obtained from the output of $\mathcal{A}_1$ instead of $\mathcal{A}_2$. The output in this experiment is:

  1. Pick $\mathbf{h}$ and $\mathbf{s}$ uniformly at random from their respective domains. Compute $z_i = \langle h_i, \mathbf{s}\rangle$ for $i \leq r$ and $z_i = \langle h_i, \mathbf{s}\rangle \oplus m_{i-r}$ for $r+1 < i \leq t \log n$.
  2. Pick $\omega_2 \leftarrow \{0,1\}^{|\omega|}$. Set $r_{\mathcal{Z}} = (\omega_2, \mathbf{s})$ and $r_{\mathcal{A}_1} = \lambda$ (the empty string). Compute

     $$(\tau, \mathsf{view}_{P_2}) \leftarrow \mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_1,\mathcal{Z}^*}(\kappa,\omega,(r_{\mathcal{Z}}, r_{\mathcal{A}_1})).$$

     Set the commitment phase transcript $\sigma = (\tau, \mathbf{h}, \mathbf{z})$.
  3. Output $\big(m, \sigma, (\omega_2, \mathbf{s}, r^*, \mathbf{h})\big)$ where $r^* = \mathsf{InvSamp}(\tau, \mathsf{view}_{P_2})$.

  The only difference between the experiments H_2 and H_3 is in Step 2. Namely, since $\mathcal{A}_1$ and $\mathcal{A}_2$ are semi-honest adversaries that output the same information, we have that

  $$\mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_2,\mathcal{Z}^*}(\kappa,\omega,(\omega_2,\cdot)) \equiv \mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_1,\mathcal{Z}^*}(\kappa,\omega,(\omega_2,\cdot)).$$

  Therefore, the outputs of H_2 and H_3 are identically distributed.

- **Hybrid** H_4: In this hybrid experiment we obtain the transcript in the commitment phase according to the Com' algorithm using simulator $\mathcal{S}_1$ but we incorporate the message m honestly. Namely,

  1. Pick $\mathbf{h}$ and $\mathbf{s}$ uniformly at random from their respective domains. Compute $z_i = \langle h_i, \mathbf{s}\rangle$ for $i \leq r$ and $z_i = \langle h_i, \mathbf{s}\rangle \oplus m_{i-r}$ for $r+1 < i \leq t \log n$.
  2. Pick $\omega_2 \leftarrow \{0,1\}^{|\omega|}, r_{\mathcal{S}_1} \leftarrow \{0,1\}^*$. Set $r_{\mathcal{Z}} = (\omega_2, \mathbf{s})$. Compute

     $$(\tau, \mathsf{view}_{P_2}) \leftarrow \mathbf{IDEAL}_{f,\mathcal{S}_1,\mathcal{Z}^*}(\kappa,\omega,(r_{\mathcal{Z}}, r_{\mathcal{S}_1})).$$

     Set the commitment phase transcript $\sigma = (\tau, \mathbf{h}, \mathbf{z})$.
  3. Output $\big(m, \sigma, (\omega_2, \mathbf{s}, r^*, \mathbf{h})\big)$ where $r^* = \mathsf{InvSamp}(\tau, \mathsf{view}_{P_2})$.

  The only difference between the experiments H_3 and H_4 is in Step 2 and indistinguishability follows directly from the indistinguishability of the simulation by $\mathcal{S}_1$ as

  $$\mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_1,\mathcal{Z}^*}(\kappa,\omega) \stackrel{\mathrm{c}}{\approx} \mathbf{IDEAL}_{f,\mathcal{S}_1,\mathcal{Z}^*}(\kappa,\omega).$$

- **Hybrid** H_5: In this experiment the commitment is generated exactly as in the previous hybrid H_4 with the exception that instead of honestly computing $\mathbf{h}, \mathbf{z}$ by first sampling $\mathbf{s}$, the experiment returns $\mathbf{h}, \mathbf{z}$ where $\mathbf{z}$ are chosen at random. Moreover, to decommit we follow the strategy of Adapt. More formally, the output in this experiment is:

1. Pick $\mathbf{h} = (h_1, \ldots, h_{t \log n})$ and $\mathbf{z} = (z_1, \ldots, z_{t \log n})$ at random from the appropriate domains. Following the Adapt algorithm, using message m compute $z^*_{r+i} = z_{r+i} \oplus m_i$ for $1 \leq i \leq ct/4$. Denote the matrix $\mathbf{A}$ where row i contains the bits of h_i. Next define the vector $\mathbf{z}^* = (z_1, \ldots, z_r, z^*_{r+1}, \ldots, z^*_{t \log n})$. We solve the set of linear equations $\mathbf{As} = \mathbf{z}^*$ to obtain $\mathbf{s}$.
2. Pick $\omega_2 \leftarrow \{0,1\}^{|\omega|}, r_{\mathcal{S}_1} \leftarrow \{0,1\}^*$. Set $r_{\mathcal{Z}} = (\omega_2, \mathbf{s})$. Compute

$$(\tau, \mathsf{view}_{P_2}) \leftarrow \mathbf{IDEAL}_{f,\mathcal{S}_1,\mathcal{Z}^*}(\kappa, \omega, (r_{\mathcal{Z}}, r_{\mathcal{S}_1})).$$

   Set the commitment phase transcript $\sigma = (\tau, \mathbf{h}, \mathbf{z})$.
3. Output $\big(m, \sigma, (\omega_2, \mathbf{s}, r^*, \mathbf{h})\big)$ where $r^* = \mathsf{InvSamp}(\tau, \mathsf{view}_{P_2})$.

By construction, the output of this experiment is identically distributed to D_2. We remark that the only difference between hybrids H_4 and H_5 is in how $(z_1, \ldots, z_{r+ct/4})$ and $\mathbf{s}$ are generated in Step 1. In H_4, we first sample $\mathbf{s}$ uniformly at random and then set $(z_1, \ldots, z_{r+ct/4})$ according to $\mathbf{s}$ and message m, whereas in H_4 we have that $\mathbf{z}$ are set to random strings and $\mathbf{s}$ is obtained later using m. Given $\mathbf{h}$, and $z_1, \ldots, z_r$ there is a bijective mapping from $\{\mathbf{s} \mid h_i(\mathbf{s}) = z_i \ \forall 1 \leq i \leq r\}$ to $\{(z_{r+1}, \ldots, z_{r+ct/4}) \mid z_{r+i} = h_{r+i}(\mathbf{s}) \ \forall 1 \leq i \leq ct/4\}$, namely the map $(\langle h_{r+1}, \mathbf{s}\rangle, \ldots, \langle h_{r+ct/4}, \mathbf{s}\rangle)$. This is because the matrix $\mathbf{A}$ where h_i occupies row i has full rank. Now, since for any message m we have that

$$\begin{aligned}\{(z_{r+1}, \ldots, z_{r+ct/4}) \mid z_{r+i} = h_{r+i}(\mathbf{s}) \forall\ 1 \leq i \leq ct/4\} \\ = \{(z_{r+1} \oplus m_1, \ldots, z_{r+ct/4} \oplus m_{ct/4}) \mid z_{r+i} = h_{r+i}(\mathbf{s}) \forall\ 1 \leq i \leq ct/4\}.\end{aligned}$$

This implies that H_4 and H_5 are identically distributed.

This concludes the proof of indistinguishability.

Perfect binding. Finally, for an invalid statement x that is not in $\mathcal{L}_{\mathcal{R}}$, we need to show no adversarial committer can equivocate a commitment to two different messages. We first show that if $x \notin \mathcal{L}_{\mathcal{R}}$, except for $2^{-O(t)}$ fraction of possible values of $\mathbf{s} = (s_1, \ldots, s_t)$, it is impossible for any (unbounded) adversary to generate the views for the servers with indices in $\Gamma = \{s_1, \ldots, s_t\}$. In essence, this is the soundness proof of the outer MPC protocol. We recall that in [IKOS09] it is proven that except for $2^{-O(t)}$ of challenge sets S no adversary can produce consistent views for the servers with indices in S. A major difference between the proof presented in [IKOS09] and what is required for our proof is in the probability distribution of S. Specifically, in [IKOS09] it was the uniform distribution over all t subsets of $[n]$. However, here the subset is generated by sampling $s_1, \ldots, s_t$ where each s_i is uniformly distributed over $[n]$. We first establish the soundness proof for our distribution.

As in the proof of [IKOS09], we consider an inconsistency graph G based on n views $V_1, \ldots, V_n$ where there is an edge between i and j if the views V_i and V_j are inconsistent. Another major difference between the [IKOS09] proof and our scenario here is that the views are not completely committed to in our case. Specifically, each server is emulated as a two-party computation between P_1 and P_2 and only P_1's view is committed in the two-party computation because its input and randomness are part of the OT invocations used for the watchlist setup. Instead, we prove "soundness" for a fixed value of P_2's input ω_2 and then conclude the proof using a union bound argument over the 2^n possible values for ω_2. As mentioned in our discussion on unique randomness, given a transcript τ, P_1's input and randomness are fixed and given the input ω_2 of P_2, the random tape of P_2 is fixed. Therefore, the views of all servers emulated by P_1 and P_2 are determined given τ, ω_2.

Next, for a fixed input ω_2 picked for P_2 we show why soundness holds for a subset Γ chosen according to the distribution of **s**. Given the transcript τ and P_2's input ω_2, let V_j be the view of server S_j for $j \in [n]$. As in the [IKOS09] approach, we consider the following two cases depending on the graph G.

Case 1: There exists a vertex cover B in G of size at most t. The idea in this case is that if all the views V_i for $i \notin B$ are consistent with each other then their output in the computation must be 0. This is because we can consider an execution of the outer protocol where the parties in B are corrupted by an adversary and behave is such as way that the views of the parties for every (honest) party $i \notin B$ results in V_i. Recall that t-robustness property of the outer protocol ensures that if the function computed by the outer protocol does not have the output 1 in its range then it is impossible for t corrupted parties to make any honest party output 1. Now, since the NP statement x is not in L, we have the 1 is not in the range of the function computed by the outer protocol. Therefore, it must follow that for every $i \notin B$, the view V_i must have the party output 0. This means that if Γ contains some index outside B then the adversary is caught. Since each index s_i assumes a value in $[n]$, the fraction of tuples $(s_1, \ldots, s_t)$ for which all $s_i \in B$ is $\left(\frac{t}{n}\right)^t = 2^{-\Omega(t)}$.

Case 2: The minimum vertex cover in G is of size greater than t. As in [IKOS09], we rely on the fact that if the minimum vertex cover is at least t, then there must exist a matching of size $\frac{t}{2}$. Therefore, we need to compute the fraction of tuples $(s_1, \ldots, s_t)$ such that there exists no i, j such that (s_i, s_j) is an edge in the matching. Let the indices of the matching edges be $(i_1, i_{t/2+1}), \ldots, (i_{t/2}, i_t)$. We now have the following claim:

Claim 6.1. *The fraction of tuples such that $|\{s_1, \ldots, s_{t/2}\} \cap \{i_1, \ldots, i_{t/2}\}| < \frac{t^2}{8n}$ is at most $e^{-\frac{t^4}{128n^2}} = 2^{-O(t^2)}$.*

Proof: Set the random variable X_j to be 1 if i_j is chosen among $s_1, \ldots, s_{t/2}$. Now, since each s_i is chosen uniformly at random we have that $\Pr[X_j = 1] = \frac{t}{2n}$. By the linearity of expectations it follows that

$$E[\sum_{j=1}^{t/2} X_i] = \frac{t^2}{4n}.$$

Using the Chernoff-Hoeffding inequality we have that

$$\Pr\left[\sum_{j=1}^{t/2} X_j \leq \frac{t^2}{8n}\right] \leq \Pr\left[\left|\sum_{j=1}^{t/2} X_j - \frac{t^2}{4n}\right| \geq \frac{t^2}{8n}\right] \leq e^{-\frac{t^4}{32n^2}}.$$

■

Now, conditioned on there being at least $\frac{t^2}{8n} = O(t)$ vertices from $i_1, \ldots, i_{t/2}$ in $s_1, \ldots, s_{t/2}$, the probability that all indices $s_{t/2+1}, \ldots, s_t$ miss the $O(t)$ matched vertices in $i_{t/2+1}, \ldots, i_t$ is at most $(1 - \frac{O(t)}{n})^{t/2} = e^{-\Omega(\frac{t^2}{n})} = 2^{-\Omega(t)}$.

We can conclude that for a fixed value of ω_2, the fraction of possible tuples that an adversary can reveal as a decommitment is at most $2^{-\Omega(t)}$. We set c to be a constant such that this fraction is bounded by 2^{-ct}. Since $t = O(n)$, we set n so that $ct > 2|\omega|$. Therefore, by using a union bound on every possible value of ω_2 we get that the total fraction of tuples for which there exists some ω_2 that can be revealed in the decommitment is bounded by $2^{-ct+|\omega|} < 2^{-ct/2}$.

Next, given a transcript τ submitted in the first message of the commitment phase, let S denote the set of possible tuples $\mathbf{s} = (s_1, \ldots, s_t)$ that can be revealed. Then, from the preceding argument we have that $|S| < 2^{t \log n - ct/2}$. We show now that m is bound to the commitment using the binding property of the interactive hashing protocol.

Recall that the binding property (see Proposition 2.1) says that if the message space is $\{0,1\}^{n_1}$ and an adversary can access a set S of at most 2^{m_1} elements, then the probability that it can decommit to two elements from S is at most $2^{-\kappa_{\mathrm{IH}}}$. Since $m_1 = t \log n$, $n_1 = t \log n - ct/2$ and $\kappa_{\mathrm{IH}} = ct/4$, it follows that except with probability $2^{-\Omega(\kappa_{\mathrm{IH}})} = 2^{-\Omega(t)}$, the adversarial committer cannot reveal two values for $\mathbf{s}$ in S. Therefore, with high probability there exists only one value for $\mathbf{s}$ it can reveal in the decommitment and there exists at most one value of m that it can reveal, since the bits of m are determined by $z_{r+i} \oplus h_{r+i}(\mathbf{s})$ (for $j \in [ct/4]$).

We conclude using a union bound that the overall probability that there is more than one possible value that the adversary can decommit is at most $2^{-\Omega(t)}$. ■

6.2 Semi-Adaptive Security of the [IPS08] Protocol

We briefly discuss why our protocol satisfies privacy against an adversary that statically corrupts P_1 and adaptively corrupts P_2 post execution. Recall that our combined protocol, P_1 and P_2 emulate an outer protocol. Simulating P_1 is easy, however, since P_2 receives as input the set of indices to $\mathbf{s}$, we require our simulation to be able to, in essence, "adaptively" simulate the views of the servers in the emulation of the outer protocol. We argue below that we will not require the outer protocol to be adaptively secure.[14]

To show this we rely on the fact that the view of P_1 does not bind the views of the servers in the emulation. Simulation can then be achieved by relying on the simulation of our inner protocol [GMW87] that is secure against a similar adversary that statically corrupts P_1 and adaptively corrupts P_2 post execution. Recall that the inner protocol ρ_j computes a reactive functionality that emulates the actions of server S_j where it proceeds in phases such that the inputs in a phase corresponding to ρ_j may depend on the outputs of all instances $\rho_1, \ldots, \rho_n$ from the previous phase. The first observation we make is that in every instance ρ_j and every phase, the inputs of server S_j are XOR-shared as inputs to P_1 and P_2 and the outputs are XOR-shared as well and the final output of the server is transmitted only to party P_2. This means that the view of P_1 for all instances does not bind the view of the servers as its input and output in every instance are simply random strings. Semi-adaptive security requires simulating a static corruption P_1 followed by an adaptive corruption of P_2 post execution. By relying on the simulation of [GMW87], we can simulate P_1 in our combined protocol, by simply providing its input and output for every instance as a random string.[15] Next, upon completion of the execution, P_2 is corrupted and upon receiving the inputs $w_2, \mathbf{s}$, the simulation must generate P_2's view. The high-level idea here is that since none of the servers views are bound we first generate the views for all servers in the outer protocol using the respective inputs ω_1 and ω_2 of P_1 and P_2. Next, we "map" the computation of server S_j to the instance ρ_j. Namely, the input and output of each computation step is the input and output of each step in the reactive computation. More precisely, given the input and output of P_1 that have been already fixed as part of simulating P_1 during the execution, we respectively XOR it with the actual input and output of the server in that step to obtain the input and output of party P_2. Then for each step we rely on the (semi-honest) adaptive simulation of the [GMW87] protocol to generate the view of P_2 in that step using the input and output. In this manner, we can obtain the view of

[14] As pointed out in [AL11], the protocol of [BGW88] is in fact not known to be adaptively secure and only admits exponential-time simulation of adaptive adversaries.

[15] The first execution, however, requires P_1's actual input which is provided and can be used directly.

P_2 for all steps in the reactive computation using ρ_j for every j.

Remark 6.2. *We remark that the proof of security of the protocols in [IPS08] against malicious parties crucially relies on the adaptive security of the outer protocol. This is because during the execution of their protocol when a malicious party deviates in an instance of the inner protocol corresponding to the actions of a particular server, the server is considered corrupted. As this can happen at any point during the protocol the security of the outer protocol is required to be secure against adaptive corruptions. In contrast, here the proof of indistinguishability relies on the simulation of semi-honest adversaries emulated by the real and fake commitment algorithm. Furthermore, we require only static corruption for the outer protocol as at the time the views of the servers are determined (by running the MPC-in-the-head for the outer protocol) the set of corrupted parties are already known. For an honest commitment, the input message determines the corrupted parties and for the fake commitment, the views are determined by the Adapt algorithm which knows the message and therefore the set of corrupted parties.*

7 Constructing Adaptive Zero-Knowledge Proofs

We describe next how to construct adaptive zero-knowledge proofs for all NP languages based on our instance-dependent commitment schemes from Sections 4, 5.1.1 and 6. For simplicity we focus on honest verifier zero-knowledge proofs, which can be transformed to zero-knowledge proof using standard tools.

7.1 Adaptive Zero-Knowledge Proofs with Soundness Error $1/2$

Let x denote a statement to be proven by the prover relative to some language $\mathcal{L}$ associated with relation $\mathcal{R}$. Then the prover generates a garbled circuit C that takes (x, ω) and outputs 1 only if $(x, \omega) \in \mathcal{R}$, and commits to this garbling and the secret key sk using the commitment scheme from Figure 2. Next, upon receiving a challenge bit b from the verifier, the prover continues as follow. If $b = 0$ then the prover decommits to the commitment of the secret key and the garbled circuit for which the verifier verifies the correctness of garbling. Else, if $b = 1$ then the prover decommits a "path" in the garbled circuit and provides an encoding for ω that evaluates the path to 1. Namely, we consider the concrete garbling construction by [Yao86, LP09] for which each evaluation induces a path of computation, where each gate evaluation requires the decryption of a single ciphertext out of four ciphertexts, where this ciphertext can be part of the decommitted information handed to the verifier when $b = 1$. The verifier then evaluates the garbling on this path and checks that the outcome if 1. We note that it is not clear how to generalize this property (where only part of the garbled circuit is decommitted) nor the following reconstructability property, for the general notion of garbled schemes.

Let $\mathsf{Garb} = (\mathsf{Grb}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ denote a garbling scheme as in Section 2.5. Then, we will require one more property that Garb should satisfy:

Reconstructability: Given any path of computation in the garbled circuit it is possible to reconstruct the rest of the garbled circuit as being honestly generated by Grb.

We note that the garbling scheme described in [LP09] meets this notion. Specifically, it is possible to initially honestly generate a pair of labels per wire without assigning their meaning, encrypting only one label per gate (known by the active key). Next, upon receiving the witness ω, the bit values associated with each label are determined, and the rest of the ciphertexts for each gate can be completed.

The formal description of our protocol can be found in Figure 6.

Adaptive Zero-Knowledge Proof for Any Language $\mathcal{L} \in \mathsf{NP}$

Building block: Instance-dependent commitment scheme Com for language $\mathcal{L}$.

Inputs: A circuit C as above and a public statement $x \in \mathcal{L}$ for both. A witness ω for the validity of x for the prover $\mathcal{P}$.

The protocol:

1. $\mathcal{P} \to \mathcal{V}$: $\mathcal{P}$ generates $(\widetilde{\mathrm{C}}, \mathbf{dk}, \mathsf{sk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C})$ and sends $\mathsf{Com}(\widetilde{\mathrm{C}}, \mathbf{dk})$ and $\mathsf{Com}(\mathsf{sk})$ to the verifier (where the commitments are computed using the real commitment algorithm).
2. $\mathcal{V} \to \mathcal{P}$: The verifier sends a random challenge bit $b \leftarrow \{0, 1\}$.
3. $\mathcal{P} \to \mathcal{V}$:
   - If $b = 0$ then the prover decommits to $\widetilde{\mathrm{C}}$, $\mathbf{dk}$ and sk. The verifier accepts if the decommitments are valid and that the garbling was honestly generated.
   - If $b = 1$ then the prover decommits to $\mathbf{dk}$ and further provides the decommitment for the encoding of ω and the path of computation in the commitment to $\widetilde{\mathrm{C}}$ that is evaluated during the computation of $\mathsf{Eval}(\widetilde{\mathrm{C}}, \widetilde{\omega})$. Namely, the prover invokes $\widetilde{\omega} := \mathsf{Enc}(\mathsf{sk}, \omega)$ and then decommits to the encoding of $\widetilde{\omega}$ within the commitment of sk (recall that this is possible due to the decomposability of the garbled scheme), as well as the path of computation. The verifier then invokes $\widetilde{y} := \mathsf{Eval}(\widetilde{\mathrm{C}}, \widetilde{\omega})$ and accepts if $\mathsf{Dec}(\mathbf{dk}, \widetilde{\omega})$ equals 1.

Figure 6: Adaptive zero-knowledge proof for any language $\mathcal{L} \in \mathsf{NP}$

Theorem 7.1. *Assume the existence of one-way permutations. Then, the protocol presented in Figure 6 is an adaptively secure honest verifier zero-knowledge proof for any language in NP with soundness error $1/2$.*

Using our instance-dependent commitment scheme from Section 4.1 we note that the communication complexity of our protocol is $O(\kappa s^2)$ where κ is the security parameter and s is the size of C.

Proof: Proving completeness is straightforward, as an honest prover always has a convincing strategy. Specifically, it can both properly decommit to a valid garbling and secret key as well as the input labels that evaluates the garbled circuit to 1. Next, proving soundness is based on the binding property of the underlying commitment schemes. Specifically, in case $x \notin \mathcal{L}$, then a corrupted prover cannot equivocate the commitment. Moreover, by the correctness property of the garbling scheme, it holds that the prover cannot answer both possible challenges. As that implies that it constructed the garbled circuit property and that it has an encoding of an input that evaluates the garbling to 1. This argument is similar to the argument made in the proof of Theorem 4.1.

To prove the zero-knowledge property we need to construct a simulator $\mathcal{S}$ that simulates the view of the (honest) verifier. More formally, simulator $\mathcal{S}$ picks a random bit b and continues as follows. In case $b = 0$ then $\mathcal{S}$ plays the role of the honest prover throughout the entire protocol. On the other hand, in case $b = 1$ then the simulator constructs a fake garbled circuit by running $\mathsf{SimGC}(1^\kappa, \mathrm{C}, \mathbf{y})$ and then commits to $[\mathsf{SimGC}(1^\kappa, \mathrm{C}, 1)]_1$ and $[\mathsf{SimGC}(1^\kappa, \mathrm{C}, 1)]_3$ using the fake commitment algorithm. Finally, it commits to $[\mathsf{SimGC}(1^\kappa, \mathrm{C}, 1)]'_2$ using the fake commitment algorithm where $[\mathsf{SimGC}(1^\kappa, \mathrm{C}, 1)]'_2$ is a complete set of input labels that involves the second outcome of the simulated garbler and randomly chosen labels of the appropriate length. Upon receiving the bit 1 from the verifier, the simulator completes the execution

as would the honest prover do, decommitting to . Then, the indistinguishability of the real and simulated views follows from the hiding property of the instance-dependent commitment scheme for $x \in \mathcal{L}$ and the privacy of the garbling scheme, where the difference between the executions is in case that $b = 1$ such that the simulator computes a simulated circuit and uses the fake commitment algorithm.

Finally, to prove adaptivity we define the randomness presented by the simulator upon corrupting the prover and receiving the witness ω for x. That is, in case $b = 1$ the simulator must present randomness demonstrating that it committed to $\widetilde{\mathrm{C}}, \mathbf{dk}$ and sk using the real commitment algorithm rather than committing to the simulated garbling using the fake algorithm. This can be achieved as follows. The simulator first reconstructs the garbled scheme, viewing the garbled circuit as honestly generated (this follows efficiently from the reconstructability property). Next, the simulator invokes the Adapt algorithm in order to generate randomness that is consistent with the reconstructed garbled circuit. By the security of the commitment scheme, the verifier's views in the real and simulated executions are computationally indistinguishable. ■

7.2 Linear-Rate Adaptive Zero-Knowledge Proofs

In the next section we will rely on the protocol of [IKOS09] to construct an adaptively secure zero-knowledge protocol with optimal efficiency, we briefly recall this protocol below. Given an NP language $\mathcal{L}$ with the corresponding relation $\mathcal{R}$, let f be the following $(n+1)$ input functionality:

$$f(x, \omega_1, \ldots, \omega_n) = \mathcal{R}(x, \omega_1 \oplus \cdots \oplus \omega_n).$$

In addition, let Π_f be an n-party protocol which realizes f with perfect correctness, perfect t-privacy against semi-honest adversary and t-robustness against malicious parties. Furthermore, we will require one more property that Π_f should satisfy:

Reconstructability: Given any set of t indices $i_1, \ldots, i_t$ and corresponding views $V_{i_1}, \ldots, V_{i_t}$ that are consistent with an execution using Π_f, then using a witness $w \in \mathcal{R}_x$ it is possible to reconstruct views for all the remaining parties, namely V_j for $j \notin \{i_1, \ldots, i_t\}$, such that $V_1, \ldots, V_n$ are pairwise consistent (with respect to x and Π_f). In essence, this set of views demonstrates an execution using honest parties.

It can be easily verified that the semi-honest protocol variant of [BGW88] satisfies this notion as it simply involves reconstructing secret shares corresponding to the Shamir secret sharing scheme. Namely, given shares of t specific parties and a secret s, we need to be able to find consistent shares for all parties that demonstrate that the shares were honestly generated from s. Nevertheless, for our protocol to be secure we need to rely on the stronger version of [BGW88] that is secure in the presence of malicious adversaries. This is because we require our protocol to satisfy t-robustness. In Appendix B we demonstrate that for a variant of the proceeding protocol presented in [AL11] we can achieve *reconstructability*. More precisely, we will rely on a protocol that secure only against $t < n/4$ corruptions, as we only require $t = O(n)$, implying that we do not need to rely on the strongest possible setting where any minority of the parties can be corrupted. Roughly speaking, we can achieve reconstructability as it only relies on being able to achieve reconstructability for a verifiable secret sharing scheme with the same parameters. In our protocol, t is set to $O(\kappa)$ and $n = O(t)$. We now describe the zero-knowledge protocol in the commitment-hybrid model.

1. The verifier picks t distinct parties $i_1, \ldots, i_t \in [n]$ and commits the indices to the prover.
2. The prover chooses $\omega_1, \ldots, \omega_n \in \{0,1\}^{|\omega|}$ at random subject to $\omega_1 \oplus \cdots \oplus \omega_n = w$. It then emulates the execution of Π_f on input $(x, \omega_1, \ldots, \omega_n)$ and prepares the views $V_1, \ldots, V_n$ of the n parties. It finally commits to each of the n views separately.

3. The verifier decommits to the indices $i_1, \ldots, i_t$ committed to in the first message.
4. The prover decommits the views of players with indices $i_1, \ldots, i_t$, namely, $V_{i_1}, \ldots, V_{i_t}$.
5. The verifier accepts if and only if the prover successfully opens t views that are consistent with each other and the players in each of these views output 1.

It was shown in [IKOS09] that this protocol is a zero-knowledge proof in the commitment-hybrid model. Informally, correctness follows the fact that all the n views can be constructed honestly given the witness ω, and therefore any t subset of parties can be convincingly revealed. Furthermore, soundness follows from the t-robustness of the protocol Π_f and the zero-knowledge requirement follows from the fact that protocol Π_f is t-private. To realize the commitment functionality we simply rely on a statistically hiding commitment scheme for the commitments made by the verifier in the first message, whereas the commitments made by the prover in the second message are realized using a statistically binding commitment scheme.

We can easily modify the same protocol to obtain an adaptive zero-knowledge proof by simply using our instance-dependent commitment scheme to compute all the commitments made by the prover. Completeness, soundness and zero-knowledge follows as before with the exception that we need to additionally show that we can admit adaptive prover corruption when simulating a malicious verifier. This essentially entails revealing all the views $V_1, \ldots, V_n$ consistently and not just the t views requested by the malicious verifier. The hardest corruption case to prove is when the prover is post execution corrupted. In this case, given the views $V_{i_1}, \ldots, V_{i_t}$ revealed in the execution, we need to reconstruct consistent views for the remaining parties. This is possible from the reconstructability of the protocol Π_f.

We next evaluate the communication complexity of our protocol. We require to commit to n views using the instance-dependent commitment scheme with equivocation parameter set to the size of the maximum view $\max_i |V_i|$. To achieve equivocation of $\hat{t}$ bits, the communication complexity of our instance-dependent commitment scheme (including decommitments) is $O(s) \cdot \mathsf{poly}(\kappa) + \hat{t} \cdot \mathsf{poly}(\kappa, d, \log s)$. Therefore, our total communication complexity of our adaptive zero-knowledge proof is

$$\sum_{i=1}^{n} \left[O(s) \cdot \mathsf{poly}(\kappa) + |V_i| \cdot \mathsf{poly}(\kappa, d, \log s) \right] = nO(s) \cdot \mathsf{poly}(\kappa) + C_{\Pi} \mathsf{poly}(\kappa, d, \log s)$$

where C_Π is the communication complexity of Π_f. Therefore, we have the following theorem.

Theorem 7.2. *Suppose that Π_f realizes functionality f with perfect t-robustness and perfect t-privacy (in the semi-honest model) where $n = ct$ for some constant $c > 1$. Then there exists an adaptive zero-knowledge proof for the NP relation $\mathcal{R}$ with soundness error 2^{-t} and communication complexity $O(st) \cdot \mathsf{poly}(\kappa) + C_{\Pi} \mathsf{poly}(\kappa, d, \log s)$ where s is the size of the circuit that verifies the NP relation and d is the depth of the circuit.*

To obtain our most efficient construction, we rely on the variant of [BGW88] presented in [AL11] combined with the modification of the circuit presented in [IKOS09]. Given an NP relation $\mathcal{R}$, this modification allows us to assume (without loss of generality) that the input to the circuit verifying the $\mathcal{R}$ is of size s where it includes the intermediate values in all the wires in the computation using the real witness ω in addition to ω itself. Such a circuit will have the size $O(s)$ and depth $O(1)$ and the corresponding communication complexity of the MPC protocol Π will be $C_\Pi = O(ns)\mathsf{poly}(\kappa)$. We plug this in our main theorem and set $t = \omega(\log \kappa)$ for negligible soundness. Therefore, we have the following corollary,

Corollary 7.3. *Assume the existence of one-way functions. Then, for any NP relation $\mathcal{R}$ that can be verified by a circuit of size s (using bounded fan-in gates), there exists an adaptive zero-knowledge proof with communication complexity $O(s) \cdot \mathsf{poly}(\kappa, \log s)$.*

One could optimize the parameters even further by relying on more efficient multiparty protocols, such as the ones presented in [DI06]. However, since committing to any message using any of our instance-dependent commitment scheme requires $\Omega(s\mathsf{poly}(\kappa))$ bits of communication, we will not be able to obtain a "constant-rate" ZK using our technique and hence do not pursue such optimizations.

References

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *FOCS*, pages 166–175, 2004.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP*, pages 152–163, 2010.

[AIKW13] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In *CRYPTO*, pages 166–184, 2013.

[AL11] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly-secure multiparty computation. *IACR Cryptology ePrint Archive*, 2011:136, 2011.

[App14] Benny Applebaum. Key-dependent message security: Generic amplification and completeness. *J. Cryptology*, 27(3):429–451, 2014.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *STOC*, pages 479–488, 1996.

[BGJ+13] Elette Boyle, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Secure computation against adaptive auxiliary information. In *CRYPTO*, pages 316–334, 2013.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[BHHI10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In *EUROCRYPT*, pages 423–444, 2010.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *CCS*, pages 784–796, 2012.

[BMO90] Mihir Bellare, Silvio Micali, and Rafail Ostrovsky. Stoc. pages 482–493, 1990.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[CCD87] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract). In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, page 462, 1987.

[CDD+04] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. Adaptive versus non-adaptive security of multi-party protocols. *J. Cryptology*, 17(3):153–207, 2004.

[CDD+15] Ignacio Cascudo, Ivan Damgård, Bernardo Machado David, Irene Giacomelli, Jesper Buus Nielsen, and Roberto Trifiletti. Additively homomorphic UC commitments with optimal amortized overhead. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 495–515, 2015.

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

[CPS+15] Michele Ciampi, Giuseppe Persiano, Alessandra Scafuro, Luisa Siniscalchi, and Ivan Visconti. Improved OR composition of sigma-protocols. *IACR Cryptology ePrint Archive*, 2015:810, 2015.

[CvdGT95] Claude Crépeau, Jeroen van de Graaf, and Alain Tapp. Committed oblivious transfer and private multi-party computation. In *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, pages 110–123, 1995.

[Dam10] Ivan Damgård. On Σ-protocols. *http://www.cs.au.dk/ ivan/Sigma.pdf*, 2010.

[DHRS04] Yan Zong Ding, Danny Harnik, Alon Rosen, and Ronen Shaltiel. Constant-round oblivious transfer in the bounded storage model. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, pages 446–472, 2004.

[DI06] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *CRYPTO*, pages 501–520, 2006.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 554–563, 1994.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO*, pages 39–56, 2008.

[GLOV12] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *FOCS*, pages 51–60, 2012.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

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

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In *CRYPTO*, pages 505–523, 2009.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. Ot-combiners via secure computation. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, pages 393–411, 2008.

[HJO+15] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. Cryptology ePrint Archive, Report 2015/1250, 2015. `http://eprint.iacr.org/`.

[HR07] Iftach Haitner and Omer Reingold. A new interactive hashing theorem. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 319–332, 2007.

[HRVW09] Iftach Haitner, Omer Reingold, Salil P. Vadhan, and Hoeteck Wee. Inaccessible entropy. In *STOC*, pages 611–620, 2009.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.

[IKOS09] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.

[IOS97] Toshiya Itoh, Yuji Ohta, and Hiroki Shizuya. A language-dependent cryptographic primitive. *J. Cryptology*, 10(1):37–50, 1997.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In *TCC*, pages 294–314, 2009.

[IW14] Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In *TCC*, pages 121–145, 2014.

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

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.

[LOP11] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In *CRYPTO*, pages 259–276, 2011.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009.

[LS90] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, pages 353–365, 1990.

[LZ11] Yehuda Lindell and Hila Zarosim. Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *J. Cryptology*, 24(4):761–799, 2011.

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

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for *NP* using any one-way permutation. *J. Cryptology*, 11(2):87–108, 1998.

[OSV15] Rafail Ostrovsky, Alessandra Scafuro, and Muthuramakrishnan Venkitasubramaniam. Resettably sound zero-knowledge arguments from owfs - the (semi) black-box way. In *TCC*, pages 345–374, 2015.

[OV08] Shien Jin Ong and Salil P. Vadhan. An equivalence between zero knowledge and commitments. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, pages 482–500, 2008.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In *ASIACRYPT*, pages 250–267, 2009.

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

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.

A Adaptive Security

In the following, we present the notion of two-party adaptive security [Can00].

Execution in the real model. Each party P_i begins with an input $x_i \in \{0,1\}^*$, a random tape r_i and the security parameter κ. An adaptive real-life adversary $\mathcal{A}$ is a probabilistic polynomial-time interactive Turing machine that starts with a random tape $r_{\mathcal{A}}$ and security parameter κ. The environment $\mathcal{Z}$ is another probabilistic polynomial-time interactive Turing machine that starts with an input z, a random tape $r_{\mathcal{Z}}$ and the security parameter κ.

At the outset of the protocol, $\mathcal{A}$ receives some initial information from $\mathcal{Z}$. Next the computation continues in rounds. Before each round, if there exists an uncorrupted party, the adversary $\mathcal{A}$ might choose to corrupt one of the parties or both. Next, $\mathcal{A}$ activates the party that is supposed to be active in this round according to the protocol. At each round, $\mathcal{A}$ sees all messages sent by the parties (that is, the conversation between the parties is visible to the adversary).

Upon corrupting a party, the adversary learns its input and its random tape. In addition, $\mathcal{Z}$ learns the identity of the corrupted party and hands some auxiliary information to $\mathcal{A}$. If the adversary is malicious, once a party is corrupted, it follows the adversarys instructions from this point. If the adversary is semi-honest, the corrupted party continues following the protocol. At the end of the computation, the parties locally generate their outputs. Uncorrupted parties output their output as specified by the protocol and corrupted parties output a special symbol $\perp$. In addition the adversary outputs an arbitrary function of its internal state. (Without loss of generality, this output consists of all the information seen in the execution: the random tape $r_{\mathcal{A}}$, the information received from the environment and the corrupted parties views of the execution.

Next, a postexecution corruption process begins. $\mathcal{Z}$ learns the outputs. Next, $\mathcal{Z}$ and $\mathcal{A}$ interact in at most two rounds, where in each round $\mathcal{Z}$ can generate a "corrupt P_1" or "corrupt P_2" message and hand it to $\mathcal{A}$. Upon receipt of this message, $\mathcal{A}$ hands $\mathcal{Z}$ the internal state of the party. At the end of this process, $\mathcal{Z}$ outputs its entire view of the interaction with the parties and $\mathcal{A}$.

Let $\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x_0, x_1, z, \mathbf{r})$ the output of $\mathcal{Z}$ on input z, random tape $r_{\mathcal{Z}}$ and a security parameter κ upon interacting with $\mathcal{A}$ and parties P_0, P_1 that engage in protocol Π on inputs $r_{\mathcal{A}}$ and $(x_0, r_0), (x_1, r_1)$, respectively, where $\mathbf{r} = (r_{\mathcal{Z}}, r_{\mathcal{A}}, r_0, r_1)$. Let $\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x_0, x_1, z)$ denote a random variable describing $\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x_0, x_1, z, \mathbf{r})$ where the random tapes are chosen uniformly. Let $\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble:

$$\{\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,\kappa\in\mathbb{N}}.$$

Execution in the ideal model. Each party P_i has input x_i and no random tape is needed. An adaptive ideal-process adversary $\mathcal{S}$ is a probabilistic polynomial-time interactive Turing machine that starts with a random tape $r_{\mathcal{S}}$ and the security parameter κ. The environment $\mathcal{Z}$ is another probabilistic polynomial-time interactive Turing machine that starts with an input z, a random tape $r_{\mathcal{Z}}$ and the security parameter κ. In addition, there is an incorruptible trusted party $\mathcal{T}$. The ideal process proceeds as follows:

First corruption phase: $\mathcal{S}$ receives some auxiliary information from $\mathcal{Z}$. Next, $\mathcal{S}$ proceeds in at most two iterations, where in each iteration $\mathcal{S}$ may decide to corrupt one of the parties. Once a party is corrupted, its input becomes known to $\mathcal{S}$. In addition, $\mathcal{Z}$ learns the identity of the corrupted party and hands some auxiliary information to $\mathcal{S}$.

Computation phase: In the semi-honest setting, uncorrupted parties forward their input to the trusted party. In the malicious setting, corrupted parties hand $\mathcal{T}$ the values chosen by $\mathcal{S}$. Let y_0, y_1 be the values

handed to $\mathcal{T}$. $\mathcal{T}$ computes $f(y_0, y_1)$ and hands P_1 the value $f(y_0, y_1)_1$ and P_2 the value $f(y_0, y_1)_2$.

Second corruption phase: $\mathcal{S}$ continues to another corruption phase, where it might choose to corrupt one of the parties based on its random tape and the information it gathered so far. Once a party is corrupted, $\mathcal{S}$ learns its input, $\mathcal{Z}$ learns the identity of the corrupted party and hands $\mathcal{S}$ some auxiliary information.

Output: Each uncorrupted party P_i outputs $f(y_0, y_1)_i$. Corrupted parties output a special symbol $\perp$. The adversary $\mathcal{S}$ outputs an arbitrary function of its internal state. $\mathcal{Z}$ learns all outputs.

Post-execution corruption phase: After the outputs are generated, $\mathcal{S}$ proceeds in at most two rounds with $\mathcal{Z}$, where in each round, $\mathcal{Z}$ can generate a "corrupt P_i" message and hand it to $\mathcal{S}$. For any such request, $\mathcal{S}$ generates some arbitrary answer and it might choose to corrupt any of the parties. The interaction continues until $\mathcal{Z}$ halts with an output.

We denote by $\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(\kappa, x_0, x_1, z, \mathbf{r})$ the output of $\mathcal{Z}$ on input z, random tape $r_{\mathcal{Z}}$ and security parameter κ upon interacting with $\mathcal{S}$ and parties P_0, P_1, running an ideal process with inputs $r_{\mathcal{S}}$ and x_0, x_1, respectively, where $\mathbf{r} = (r_{\mathcal{Z}}, r_{\mathcal{S}})$. Let $\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(\kappa, x_0, x_1, z)$ denote a random variable describing $\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(\kappa, x_0, x_1, z, \mathbf{r})$ when the random tapes $r_{\mathcal{Z}}$ and $r_{\mathcal{S}}$ are chosen uniformly. Let $\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}$ denote the distribution ensemble:

$$\{\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(\kappa, x_0, x_1, z)\}_{x_0,x_1,z\in\{0,1\}^*,\kappa\in\mathbb{N}}$$

Then we define security as follows.

Definition A.1. *Let* Π *be a protocol computing a functionality* f*. We say that* Π securely computes the functionality f in the presence of adaptive semi-honest/malicious adversaries *if for every probabilistic polynomial-time adaptive semi-honest/malicious real-life adversary* $\mathcal{A}$ *and for every environment* $\mathcal{Z}$*, there exists a probabilistic polynomial-time semi-honest/malicious ideal adversary* $\mathcal{S}$*, such that:*

$$\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\approx} \mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}.$$

Adaptive zero-knowledge. As explained in [LZ11], when considering zero-knowledge as a special case of secure computation, it is most natural to define an adaptive zero knowledge proof of knowledge functionality of the form $\mathcal{F}_{\mathcal{R}}((x, \omega), \lambda) \mapsto (-, (x, b))$ where $b = 1$ if $\mathcal{R}(x, \omega) = 1$ and $b = 0$ if $\mathcal{R}(x, \omega) = 0$. However, since the goal here is to design adaptive zero-knowledge Lindell and Zarosim considered a simplified definition that is more in line with the standard setting of zero-knowledge proof systems (that are not necessarily proofs of knowledge).

Recall that in the standard setting of zero-knowledge, indistinguishability of the real world from the ideal world is only required for instances $x \in \mathcal{L}$. For these instances the trusted party always returns 1, and therefore the trusted party can be omitted from the ideal world.

In this case the real-life model is as defined above where the input of the verifier is an instance $x \in \{0,1\}^\kappa$ (where κ is the security parameter) and the input of the prover is a pair $(x, \omega) \in \{0,1\}^\kappa \times \{0,1\}^{p(\kappa)}$ for a polynomial $p(\cdot)$. The output of the uncorrupted prover is an empty string and the output of the uncorrupted verifier is a bit specified by the protocol.

In the ideal process, the ideal process adversary $\mathcal{S}$ receives the instance x that is guaranteed to be in the language as input and interacts with the environment and corrupted parties. Thus, only 3 stages are needed: first corruption stage, output stage and postexecution corruption stage (since there is no computation stage, there is also no need for a second corruption stage).

The distribution $\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ denotes the distribution ensemble

$$\{\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x, \omega, z)\}_{x\in\mathcal{L},\omega\in\mathcal{R}_x z\in\{0,1\}^*,\kappa\in\mathbb{N}}$$

and $\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}$ denote the distribution ensemble:

$$\{\mathbf{IDEAL}^{\text{ZK}}_{f,\mathcal{S},\mathcal{Z}}(\kappa, x, \omega, z)\}_{x\in\mathcal{L},\omega\in\mathcal{R}_x z\in\{0,1\}^*,\kappa\in\mathbb{N}}.$$

Definition A.2. *Let $\mathcal{L}$ be a language. We say that $\langle\mathcal{P},\mathcal{V}\rangle$ is an adaptive zero-knowledge proof system (AZK) for $\mathcal{L}$ if $\langle\mathcal{P},\mathcal{V}\rangle$ is an interactive proof system for $\mathcal{L}$ and for any* PPT*real-life adversary $\mathcal{A}$ and any* PPT *environment $\mathcal{Z}$, there exists a probabilistic* PPT *adaptive ideal-process adversary $\mathcal{S}$, such that*

$$\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{\text{c}}{\approx} \mathbf{IDEAL}^{\text{ZK}}_{f,\mathcal{S},\mathcal{Z}}.$$

B On Reconstructability of the [BGW88] Scheme for $\mathbf{n \geq 4t + 1}$

In this Section, we argue the reconstructability of the [BGW88] scheme. We will use the variant of the [BGW88] scheme that allows *active* corruptions of up to $t < n/4$ parties. The reason for relying on this variant is that it admits a simpler protocol for computing the "product" of shared values in the presence of malicious adversaries, as shown in [AL11]. Roughly speaking, when $n \geq 4t+1$ it is possible to leverage the fact that a Reed Solomon code with parameters $(n = 4t + 1, k = 2t + 1, d = 2t + 1)$ allows the correction of up to t errors.

We briefly recall the [BGW88] protocol at a high-level. In an initial phase, all parties share their private inputs using a t-out-of-n *verifiable secret sharing* (VSS) protocol. They then evaluate the circuit gate by gate, where given the shares for the input wires of a gate they compute the shares of the output. For every addition gate, this computation simply entails each party adding their shares locally. Multiplication on the other hand requires more effort. Typically the shares all lie on a polynomial of degree t in some finite field and this property needs to be maintained. For addition gates, adding the shares maintains this property. The same property holds for multiplication-by-a-constant gates. In essence, any linear function can be evaluated this way, where after the initial phase all computation are local, and in the final step the parties exchange the shares of the output wires and each party reconstructs the output.

However, when multiplying two shared values by simply multiplying the corresponding shares, then the resulting shares would lie on a polynomial of degree $2t$. Nevertheless, when considering only passive adversaries the parties can locally multiply their shares and then execute a "degree-reduction" step. In this step the parties jointly evaluate $A \times s$, where A is a fixed matrix and s is the vector of shares, to obtain shares that will lie on a polynomial of degree t and reconstruct to the same secret as s. In essence, using the shares as private inputs, the parties compute the linear function $A \cdot s$. As mentioned before such functions can be computed securely (even in the active security case).

The approach needed for multiplication gates in the malicious case is slightly modified. It suffices to argue how one can compute a matrix multiplication on the shares. As in the passive case, the parties first locally compute the product of their shares and then secret share these products using a VSS scheme. They then multiply the local shares with A to obtain shares of the output for which they share with all the parties. Given the shares of each coordinate in the output each party tries to reconstruct the output. In the passive case, this is simple as all parties are semi-honest. However, in the malicious case there could be up to t incorrect shares for each coordinate in the output vector. In this case, the parties rely on Reed-Solomon codes to correct the errors. A subtle point here is that the initial shares are themselves products of shares that

lie on t-degree polynomial which means that the product lies on $2t$-degree polynomial. Hence, one needs to verify if the error correction threshold is maintained to obtain the Reed-Solomon code. Specifically, since there are $n = 4t + 1$ shares of a Reed Solomon code with distance $2t + 1$, the maximum number of errors that can be corrected is $\frac{4t+1-(2t+1)}{2} = t$ errors which is the maximum number of corruptions and therefore the protocol follows.

Given this variant of the [BGW88] protocol, we argue that it satisfies reconstructability. Recall that reconstructability requires that given any set of t indices $i_1, \ldots, i_t$ and the corresponding views $V_{i_1}, \ldots, V_{i_t}$ that are consistent within an execution of protocol Π_f, then using a witness $w \in \mathcal{R}_x$ it is possible to reconstruct the views for all the remaining parties. Namely, V_j for $j \notin \{i_1, \ldots, i_t\}$, such that $V_1, \ldots, V_n$ are pairwise consistent (with respect to x and Π_f). In essence, this set of remaining views demonstrates an execution using honest parties. We remark that reconstructability as stated here is simpler than a more general variant that requires to generate views of the remaining honest parties even after t parties have been maliciously corrupted. In our case, the inital t views are views of parties following the protocol honestly.

Given the witness w we observe that it is possible to obtain the actual values transmitted in every wire in the circuit by simply computing the circuit. Now, since the computation proceeds gate by gate, we can think of the view V_i as comprising of sub-views describing the messages exchanged in the computation of each gate. The (sub-)view for each gate for party P_i contains its input, which are the shares corresponding to the values of the input wires, the shares corresponding to the value in the output wire and the messages exchanged between the parties. If the output of a gate is fed as input to a subsequent gate, the share is carried forward. Therefore, it suffices to show that it is possible to extend the (sub-)view corresponding to each gate in a consistent way to the remaining parties. We demonstrate this for each gate where we show that if we have the output shares corresponding to all parties, and the actual values for the input and output wires of that gate, then we can generate the inputs shares to all parties, that will be consistent with an honest behavior and with views $V_{i_1}, \ldots, V_{i_t}$. Then since the output shares for all parties corresponding to each output wires are in each view V_{i_j} we can backtrack gate by gate and obtain input shares for each wire. This process ends with giving input shares to all parties corresponding to the input wires of the circuit. The shares corresponding to the input wires were computed using the sharing phase of the VSS protocol. To complete the reconstructability we show that the views from the VSS sharing phase can be extended as well.

It follows from the preceeding discussion that it suffices to demonstrate reconstructability for addition gates, multiplication gates and the VSS secret sharing protocol. We provide the reconstruction procedure without arguing correctness. Given the actual values of each wire and the views of the t parties the reconstruction is unique. There are special values $\alpha_1, \ldots, \alpha_n$ in the field $\mathbb{F}$ where α_i is associated with player P_i. All parties know this value corresponding to every other player.

Reconstructability of addition gates: Recall that the addition gates are computed internally by simply adding the shares corresponding the input wires. Let the output value be c and the two input values be a and b where $a + b = c$. In the view V_{i_j} for computing an addition gate, there will be the input shares (a_{i_j}, b_{i_j}) corresponding to the two input wires and all shares $(c_1, \ldots, c_n)$ corresponding to the output wire that reconstruct to the value c. We need to extend the views of the t parties to the remaining parties. This is easily achieved by first computing the unique degree-t polynomial $g_1(x)$ such that $g_1(\alpha_{i_j}) = a_{i_j}$ for all $j \in [t]$ and $g_1(0) = a$ and then setting the input shares corresponding a for the remaining parties as $g_1(\alpha_j)$ for party P_j. Analogously, $g_2(x)$ can be computed for input wire carrying the value b. It will follow that if c_i's were computed correctly, i.e. corresponding to some polynomial $g_3(x)$ such that $g_3(\alpha_j) = c_j$ for every j, then reconstruction will be correct.

Reconstructability of the VSS secret sharing protocol. We need to show that given the views of parties $P_{i_1}, \ldots, P_{i_t}$ for the secret-sharing phase of the VSS protocol we can reconstruct the views of the parties. We describe the protocol below and deliberately leave the conflict resolution part as we require to only demonstrate reconstruction when we have views consistent with honest parties.

Recall that in the VSS protocol, party P_i proceeds as follows:

1. It receives two polynomials $f_i(x)$ and $g_i(x)$ from the Dealer.
2. It sends $f_i(\alpha_j)$ and $g_i(\alpha_j)$ to P_j for all $j \in [n]$.
3. It outputs $f_i(0)$ as its share.

The functions f_i, g_i are chosen by the Dealer so as to satisfy $S(x, \alpha_i)$ is the function $f_i(x)$ and $S(\alpha_i, x)$ is the function $g_i(x)$ for every i, where $S(x, y)$ is a bivariate polynomial in degree t and $S(0, z)$ is the degree t polynomial $q(z)$ where $q(0)$ is the secret s.

The view of party P_{i_j} contains $f_{i_j}(x), g_{i_j}(x), (f_1(\alpha_{i_j}), \ldots, f_n(\alpha_{i_j}))$ and $(g_1(\alpha_{i_j}), \ldots, g_n(\alpha_{i_j}))$. Given $f_{i_1}(0), \ldots, f_{i_t}(0)$ and the secret s, it is possible reconstruct the degree-t polynomial $q(z)$ as $q(i) = f_i(0)$ and $q(0) = s$. Given the t polynomials $f_{i_1}(x), \ldots, f_{i_t}(x)$ and $q(z)$ it is possible to reconstruct the unique t-degree bivariate polynomial S. From S we can reconstruct the view of all parties.

Reconstructability of multiplication gates: As mentioned before, multiplication involves the following steps for party P_j with input shares a_j, b_j:

1. Compute $c_j = a_j \cdot b_j$.
2. Use the VSS secret sharing protocol to share c_j with all parties. Let $g_j(x)$ be the degree-$2t$ polynomial where P_i receives $g_j(\alpha_i)$ and $g_j(0) = c_j$. Notice that the secret sharing uses a degree $2t$ instead of t. This is because the values c_j lie on a degree $2t$ polynomial if the a_j's and b_j's lie on a degree t polynomial.
3. After receiving $g_1(\alpha_j), \ldots, g_n(\alpha_j)$, compute $A \cdot \mathbf{v}_j$ were $\mathbf{v}_j = (g_1(\alpha_j), \ldots, g_n(\alpha_j))$ and send it to all parties.
4. After receiving $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$, apply Reed Solomon decoding to correct all errors in vectors $\mathbf{u}_i$ for every i where $\mathbf{u}_i$ is the vector containing the i^{th} entries of $\mathbf{v}_1, \ldots, \mathbf{v}_n$ (in other words, it is the i^{th} column in the matrix B where B contains $\mathbf{v}_j$ in the j^{th} row). Let the corrected vectors be $\mathbf{u}_i^*$ for every $i \in [n]$.
5. Apply the share-reconstruction to each vector $\mathbf{u}_i^*$ to obtain c_i and then apply the share-reconstruction once again on $(c_1, \ldots, c_n)$ to obtain c.

From the preceding description, we have that the view of party P_{i_j} contains $a_{i_j}, b_{i_j}, (g_1(\alpha_{i_j}), \ldots, g_n(\alpha_{i_j}))$, $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ and c the value in the output wire. Given the vectors $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ and A it is possible to obtain $g_i(\alpha_j)$ for every i and j by simply computing $A^{-1} \cdot B$ and checking the entry (i, j) where recall B is the matrix with $\mathbf{v}_i$ in the i^{th} row. These values will define unique polynomials g_j for the remaining $n - t$ parties from which c_j values can be computed as $g_j(0)$. Next given a_{i_j} and b_{i_j} for every $j \in [t]$ and the actual values in the wires a and b we can compute the remaining shares a_j and b_j as in the case with addition by constructing the unique degree t polynomials g and g' such that $g(\alpha_{i_j}) = a_{i_j}$ and $g'(\alpha_{i_j}) = b_{i_j}$ for every $j \in [t]$ and $g(0) = a$ and $g'(0) = b$.

We remark here that the Reed Solomon decoding will always result in a syndrome of all 0's vector with no errors as reconstruction is required only when all parties are honest. Therefore, producing a view for the decoding procedure will be similar for the passive adversary case where we reconstruct a t-degree polynomial and output the value at 0.