J Cryptol (2020) 33:271–318
https://doi.org/10.1007/s00145-019-09314-2

Journal of
CRYPTOLOGY

On the Power of Secure Two-Party Computation

Carmit Hazay
Bar-Ilan University, Ramat Gan, Israel
carmit.hazay@biu.ac.il

Muthuramakrishnan Venkitasubramaniam
University of Rochester, Rochester, NY14611, USA
muthuv@cs.rochester.edu

Communicated by Nigel Smart.

Received 7 January 2018 / Revised 27 January 2019
Online publication 7 February 2019

Abstract. Ishai, Kushilevitz, Ostrovsky and Sahai (STOC 2007; SIAM J Comput 39(3):1121–1152, 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 model 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 2009; J Cryptol 24(4):761–799, 2011)]. As an application of our techniques, we show how to obtain a ZK proof with an "input-delayed" property for any NP language without relying on expensive Karp reductions that is black box in the underlying one-way function. Namely, the input-delayed property allows the honest prover's algorithm to receive the actual statement to be proved only in the final round. We further generalize this to obtain a "commit-and-prove" protocol with the same property where the prover commits to a witness w in the second message and proves a statement x regarding the witness w in zero-knowledge where the statement is determined only in the last round. This improves

Carmit Hazay: Research was partially supported by the European Research Council under the ERC consolidators Grant Agreement No. 615172 (HIPS), and by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Bureau in the Prime Minister's Office. Research partially supported by a grant from the Israel Ministry of Science and Technology (Grant No. 3-10883). Muthuramakrishnan Venkitasubramaniam: Research supported by Google Faculty Research Grant and NSF Award CNS-1526377.

a previous construction of Lapidot and Shamir (Crypto 1990) that was designed specifically for the Graph Hamiltonicity problem and relied on the underlying primitives in a non-black-box way. Additionally, 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) from one-way functions. We show that if the 2PC protocol has mild adaptive security guarantees (which are satisfied by both the Yao's and GMW's protocol), then the resulting randomized encoding can be decomposed to an offline/online encoding.

Keywords. Adaptive zero-knowledge proofs, Secure two-party computation, Randomized encoding, Instance-dependent commitments.

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) [10,16,40,69] 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 versus 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 versus malicious adversary The adversary may be *semi-honest*, meaning that it follows the protocol specification but tries to learn more than allowed from the view, 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 an arbitrary malicious adversary.

[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 [24,40,54].

Static versus 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 [17,20,54,60]. Achieving efficiency seems also to be much harder.

Zero-knowledge Zero-knowledge (ZK) interactive protocols [39] 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 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 versus 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 parties.

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 [51] 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 [26] 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 [50,51] 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 [54], secure computation of arithmetic circuits [55], non-malleable commitments [37], zero-knowledge PCPs [56], resettably sound ZK [65] to name a few, as well as efficient protocols, such as oblivious-transfer based cryptography [44,54,55] and homomorphic UC commitments [18]. More recently, highly efficient zero-knowledge arguments with practical implementations using the MPC-in-the-head have been demonstrated [1,19,38].

In contrast, in the adaptive setting, constructing adaptive zero-knowledge proofs is significantly harder and considerably less efficient. Beaver [9] showed that unless the polynomial hierarchy collapses the ZK proof of [39] is not secure in the presence of adaptive adversaries. Quite remarkably, Lindell and Zarosim showed in [63] 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. A first motivation for our work is the goal of finding alternative approaches of constructing (efficient) adaptive ZK proofs without relying on the expensive Karp reduction step.

Randomized encoding (RE) The third fundamental primitive considered in this work is *randomized encoding* (RE). Formalized in the works of [3,48,49], 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 [69]. Additional variants have been considered in the literature in the early works of [29,59]. Since its introduction, randomized encoding has found numerous applications, especially in parallel cryptography where encodings with small parallel complexity yields highly efficient secure computation [3,48,49]. (See also [4,7,11,12,32,33,36] for other applications).

Statistical versus computational Randomized encodings can be statistical or computational depending on how close the sampled distribution is to the real distribution of $\widehat{f}$. While statistical randomized encodings exist for functions computable by NC^1 circuits, only computational REs are known for general polynomial-time computable function. We refer the reader to [5] for a more detailed investigation on the class of languages that have statistical REs.

Online/offline complexity In an online/offline setting [6], one considers an encoding $\widehat{f}(x,r)$ which can be split as an offline part $\widehat{f}_{\mathrm{OFF}}(r)$ which only depends on the function f, and an online part $\widehat{f}_{\mathrm{ON}}(x,r)$ 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, $\widehat{f}_{\mathrm{OFF}}(r)$ is published or transmitted to a cloud, and later in an online phase, the weak device upon observing the sample x, transmits the encoding $\widehat{f}_{\mathrm{ON}}(x,r)$. 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 $\widehat{f}_{\mathrm{ON}}(x,r)$. In the classic garbled circuit construction, the online complexity is proportional to $|x| \cdot \mathsf{poly}(\kappa)$ where κ is the security parameter. More recently, Applebaum, Ishai, Kushilevitz and Waters showed in [6] 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 [36] or analyzed in the presence of the so-called *programmable random-oracle* model [6]. In fact, it was shown in [6] that any adaptively secure randomized encoding must have an online complexity proportional to the output length of the function. Recently, the work of Hemenway et al. [45] provided the first construction of adaptively secure RE based on the minimal assumption of one-way functions where the online complexity is only proportional to the width of the circuit. Other works made progress toward solving this problem [57,58], yet it is still open.

While the connection between RE and secure computation has been explored only in one direction, where efficient RE yields 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 black-box use of a two-party protocol Π_f^{OT},[3] carried out between parties P_1 and P_2 for a related functionality f in the oblivious-transfer (OT) hybrid model,[4] and a commitment scheme.[5] We will require Π_f^{OT} to achieve perfect (UC) security in the presence of static semi-honest corruptions. For example, the classic protocol by Goldreich, Micali and Wigderson (GMW) [40] satisfies these requirements. We further demonstrate a variant of this transformation that yields the first zero-knowledge proof that additionally has an "input-delayed" property [22,62] and continues to rely on the underlying protocol in a black-box way. The underlying two-party protocol for this transformation will, however, require slightly stronger guarantees.
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 black-box use of a two-party protocol Π_f^{OT} carried out between parties P_1 and P_2,[6] for a related functionality f in the

[3]The functionality f can be efficiently defined by making only a black-box (oracle) access to the NP relation $\mathcal{R}$. This notion is formalized as an "oracle call" to a protocol in [52].

[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]To obtain a proof, we will be able to instantiate our commitment schemes using a statistically binding commitment scheme [64] for commitments made by the prover in the ZK protocol, and by a statistically hiding commitment scheme for commitments made by the verifier. Both these schemes can be instantiated from one-way functions [47,64].

[6]By "black-box" use of a protocol, we mean that the next-message function of the resulting protocol uses the next-message function of the underlying protocol as an oracle. However, it could be the case that the underlying protocol might depend on the implemented functionality in a non-black-box manner. This notion is formalized and explored in [52].

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 static corruption of P_1 and an adaptive corruption of P_2,[7] 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 oblivious sampleability; see Sects. 3.5 and 3.6 for more details) which will be satisfied by the standard Yao's protocol [69] based on garbled circuits.

3. A general construction of a *randomized encoding* for any function f that makes black-box use (a la [51]) 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 adaptive corruption of P_2, we obtain an online/offline RE. Moreover, our construction makes 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.

A few remarks are in order.

Remark 1.1. In transformations 2 and 3, we require the underlying 2PC protocol to only be semi-adaptive with fixed roles (where the sender is statically corrupted, whereas the receiver is adaptively corrupted). This security notion is a weak requirement and almost all known protocols that are secure in the static setting are also semi-adaptive secure. Namely, the 2PC protocols based on [69] and [40] are semi-adaptive secure in this sense; we discuss these details in Sects. 5.1 and 5.2.

Remark 1.2. Our online/offline RE based on (semi-adaptive) 2PC protocols is efficient only for certain protocols. Looking ahead, the offline complexity of the resulting RE is proportional to the honest algorithm of party P_1 and the online complexity is proportional to the semi-adaptive simulation of party P_2. In the case of [69]'s protocol, applying our transformation yields the standard RE based on garbled circuits. We note that while we do not obtain any new constructions of RE, our transformation demonstrates the relationship between the semi-adaptive simulation complexity of a protocol and the efficiency of a corresponding RE.

Comparison with [51] We remark that the approach of [51] that transforms general MPC protocols cannot be used "directly" to yield our first result concerning static ZK. This is

[7] The security notion in which one party is statically corrupted, whereas the second party is adaptively corrupted is known by semi-adaptive security [43].

because all constructions presented in their work require to instantiate the MPC protocol with at least three parties. In subsequent work, Ishai et al. [52] show how to extend the [51] transformation to obtain a result analogous to our first result. In comparison with ours, their transformation yields a more communication efficient zero-knowledge proof. Nevertheless, the approaches of [51] and [52] cannot yield the stronger "input-delayed" property as all their protocols start with an MPC protocol where the views of all parties are committed to by the prover in the first round and there is no mechanism to equivocate the views, which is required in order to obtain the input-delayed property.

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.

Commit-and-prove input-delayed ZK proofs. In [62], Lapidot and Shamir provided a three-round witness-indistinguishable (WI) proof 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 [21] 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 general constructions of input-delayed zero-knowledge proofs that yield an efficient version of the protocol of [62] for arbitrary NP-relations. We remark that our work is stronger than [21] in that it achieves the stronger adaptive soundness property (which is satisfied by [30,62]). The communication complexity in our protocol depends only linearly 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 the recent work of Hemenway et al. [45].

The "commit-and-prove" paradigm considers a prover that first commits to a witness w and then, in a second phase upon receiving a statement x asserts whether a particular relation $R(x, w) = 1$ without revealing the committed value. This paradigm, which is implicit in the work of [40] and later formalized in [20], is a powerful mechanism to strengthen semi-honest secure protocols to maliciously secure ones. The MPC-in-the-head approach of [51] shows how to obtain a commit-and-prove protocol based on one-way functions that relies on the underlying primitives in a black-box way. This technique has been used extensively in several works to close the gap between black-box and non-black-box constructions relying on one-way functions (see [37,42,65] for a few examples).

We show that our input-delayed ZK proof further supports the commit-and-prove paradigm which is additionally black box in the underlying one-way functions. More precisely, we obtain a black-box construction of a 6-round commit-and-prove protocol with the input-delayed property.

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. 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 which makes black-box access to the NP relation. Prior constructions were known only for Σ-protocols [25] and for Blum's Graph Hamiltonicity [31].

1.3. *Perspective*

Our work is similar in spirit to the work of [51,54] that demonstrated the power of information-theoretic MPC protocols in constructing statically secure protocols. Here, we show the power of (adaptively secure) 2PC protocols in the OT-hybrid in constructing adaptively secure protocols and randomized encodings. Instantiating our 2PC with the standard protocols of [69] and [40] 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.

A second contribution of our construction shows a useful class of applications for which 2PC protocols can be used to reduce the round complexity of black-box constructions. The well-known and powerful "MPC-in-the-head" technique has found extensive applications in obtaining only black-box usage of the underlying primitives (and in the case of zero-knowledge even black box in the underlying NP-relation). In many cases, their approach was used to close the gap between black-box and non-black-box constructions. In particular, their approach provided the first mechanism to obtain a commit-and-prove protocol that depended on the underlying commitment in a black-box way. We believe that our technique yields an analogous "2PC-in-the-head" technique which, in addition to admitting similar commit-and-prove protocols, can improve the round complexity as demonstrated for various cryptographic primitives. This is because the additionally input-delayed property allows the commit-and-prove protocol to run in parallel with an external protocol.

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 [63] adaptive ZK proof and the input-delayed protocol from [62] 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).

1.4. *Concurrent and Subsequent Work*

In concurrent work, Ishai et al. extend the transformation of [50] to construct zero-knowledge proofs starting from semi-honest two-party computation [52]. Their transformation yields better asymptotic communication complexity and can incorporate protocols in OT and oblivious linear evaluation (OLE) hybrid models. However, their constructions do not extend to adaptive corruptions. In subsequent work, Canetti, Poburinnaya and Venkitasubramaniam introduce functionally equivocal encryption scheme to design round optimal secure multiparty computation against adaptive corruptions [23]. They realize this primitive based on one-way functions and were inspired by the adaptive instance-dependent commitment scheme developed in this work. Finally, Ganesh et al. design (private-coin) adaptive zero-knowledge arguments based on oblivious transfer where the communication is linear in the size of the statement [35].

2. Overview of Techniques

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 [69] and [40] protocols will require the existence of an oblivious-transfer protocol. We modify the construction to rely only on one-way functions in two steps. First, we construct a randomized encoding with certain special properties. Then, we use the randomized encoding to construct a zero-knowledge protocol. The key insight that allows us to incorporate calls to the OT transfer is the following:

- For every OT call where P_1's input is (s_0, s_1) and P_2's input is t, the prover will commit to s_0 and s_1 using a statistically binding commitment scheme com (which can be based on one-way functions), in the first round.[8] Then, opening P_1's view requires decommitting both the commitments, and opening P_2's view will be accomplished by only decommitting s_b where b is receiver's input for that OT call.

We remark that our ZK proof does not provide efficiency gains compared to [51,54] (using OT-preprocessing) as we require a commitment for every oblivious-transfer and in the case of compiling [40] results in $O(s)$ commitments where s is the size of the circuit. Nevertheless, we believe that this compilation illustrates the simplicity of obtaining a ZK proof starting from any 2PC protocol.

[8]Note that in Naor's statistically binding commitment scheme [64] 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 [63] that showed that constructing adaptively secure ZK proofs can be reduced to constructing *adaptive instance-dependent commitment* schemes [13,53,63,66]. In fact, by simply instantiating the commitments from the prover in the (static) ZK proofs of [51] 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 (i.e., transcript can be simulated indistinguishably). 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 [43]. 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 be achieved 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 security against 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 that is similar to our previous transformation in order 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 [69] or [40] to obtain an adaptive instance-dependent commitment scheme. Both commitments result in a communication complexity of $O(s \cdot \mathsf{poly}(\kappa))$ 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 [51] zero-knowledge (ZK) construction, where it commits to the views of the underlying MPC protocol. The resulting protocol will have a complexity of $O(s^2 \cdot \mathsf{poly}(\kappa))$ 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.

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 $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ as defined for the previous construction. 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 $\mathcal{S}_3$ 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 protocol. 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

$\mathcal{S}_2$ statically 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-adaptive simulation of P_2. Moreover, this protocol can be instantiated by the [69] and [40] protocols, where the OT sub-protocols are implemented using one-way functions as specified before. The [69]-based protocol will result in an offline complexity of $O(s \cdot \mathsf{poly}(\kappa))$ and an online complexity of $O(n \cdot \mathsf{poly}(\kappa))$ where s is the size of the circuit implementing f and n is the input length,[9] whereas the [40] protocol will result in an offline and online complexities of $O(s \cdot \mathsf{poly}(\kappa))$. 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 preceding 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 "oblivious sampling" property. Namely, the ability to explain real randomized encoding as a simulated one.[10] Since adaptive instance-dependent commitment schemes are sufficient to realize adaptive ZK, this provides a transformation from RE to adaptive ZK.

3. 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.

[9] We note that the online complexity can be improved by relying on the work of [6].

[10] This notion has been considered in the past in the context of oblivious public-key encryption schemes requiring the ability to sample a public-key without knowing the secret key or sampling a ciphertext without the knowledge of the plaintext [27], and to switch from a real to an oblivious object.

Definition 3.1. 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 3.2. 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 $SD(X_\kappa, Y_\kappa) \leq \varepsilon(\kappa)$. 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.

3.1. *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 3.3. (*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 S$^*(z)$ using Com, is negligible.

We refer the reader to [41] 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 [64]. We further consider *pseudorandom* commitments for which the honest sender's messages in the commitment phase are pseudorandom, i.e., 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 [64] and with perfect binding based on one-way permutations.

3.2. *The Commitment Functionality*

The formal description of functionality $\mathcal{F}_{\text{COM}}$ is depicted in Fig. 1.

3.3. *Adaptive Instance-Dependent Commitment Schemes [63]*

We extend the instance-dependent commitment scheme definition of [63], originally introduced for the binary message space, to 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 $m \in \mathcal{M}_n$, an instance $x \in \{0,1\}^*$, $|x| = n$ 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 $m \in \mathcal{M}_n$, $\mathsf{Adapt}(x, \omega, c, m, r)$ 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

Functionality $\mathcal{F}_{\text{COM}}$

Functionality $\mathcal{F}_{\text{COM}}$ communicates with sender S and receiver R, and adversary $\mathcal{S}$.

1. Upon receiving input $(\mathsf{commit}, sid, m)$ from S where $m \in \{0,1\}^t$, internally record (sid, m) and send message $(sid, \mathrm{S}, \mathrm{R})$ to the adversary. Upon receiving approve from the adversary send sid, to R. Ignore subsequent $(\mathsf{commit}, ., ., .)$ messages.
2. Upon receiving (reveal, sid) from S, where a tuple (sid, m) is recorded, send message m to adversary $\mathcal{S}$ and R. Otherwise, ignore.

Fig. 1. The string commitment functionality.

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 3.4. (*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: The ensembles $\{\mathsf{Com}(x, m)\}_{x\in\mathcal{L},m\{0,1\}^{|x|}}$ and $\{\mathsf{Com}'(x)\}_{x\in\mathcal{L}}$ are computationally indistinguishable.
2. Adaptivity: The distributions $\{\mathsf{Com}(x, m; U_{p(|x|)}), m, U_{p(|x|)}\}_{x\in\mathcal{L},\omega\in\mathcal{R}_{\mathcal{L}},m\in\{0,1\}^{|x|}}$ and $\{\mathsf{Com}'(x; U_{p(|x|)}), m, \mathsf{Adapt}(x, \omega, \mathsf{Com}'(x; U_{p(|x|)}), m)\}_{x\in\mathcal{L},\omega\in\mathcal{R}_{\mathcal{L}},m\in\{0,1\}^{|x|}}$ 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 $x \notin \mathcal{L}$, $m, m' \in \mathcal{M}_{|x|}$, 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.

In an instance-dependent commitment scheme, there is no Com' algorithm and the adaptivity property is not required to hold.

Definition 3.5. (*Instance-dependent trapdoor commitment schemes*) Trapdoor commitment schemes were introduced in [8] and have been extensively used in the design of zero-knowledge proofs, non-malleable commitments, signatures, etc. In this work, we consider instance-dependent trapdoor commitment schemes, introduced by [21], that are weaker than adaptive instance-dependent commitment schemes. Let $\mathsf{Com} = (\mathrm{S}, \mathrm{R})$ be a statistically binding instance-dependent commitment scheme, $\mathcal{R}$, an NP relation and $\mathcal{L}$, the language associated with $\mathcal{R}$. We say that Com is an instance-dependent trapdoor commitment scheme, if there exists an expected PPT oracle machine $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that for any PPT R^*, $(x, w) \in \mathcal{R}$, $m \in \{0, 1\}^\kappa$, the output (τ, decom) of the following experiments is computationally indistinguishable:

– an honest sender S interacts with R^* on instance x to commit to m, and then opens the commitment: τ is the view of R^* in the commit phase, and decom 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 $(\tau, state) \leftarrow \mathcal{S}_1^{\mathrm{R}^*}(1^\kappa, x)$, $\mathsf{decom} \leftarrow \mathcal{S}_2(state, w, m)$.

3.4. *Zero-Knowledge Proofs*

Definition 3.6. (*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 3.7. (*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 denotes the output of $\mathcal{V}^*$ after it interacts with $\mathcal{P}$ on common input x, whereas the right term denotes the output of $\mathcal{S}$ on x.

Definition 3.8. (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.

Adaptive zero-knowledge This notion considers the case for which the prover is adaptively corrupted. Loosely speaking, the simulator obtains a statement $x \in \mathcal{L}$. Moreover, at any point of the execution, the adaptive adversary is allowed to corrupt the prover. It is then required that zero-knowledge holds even in the presence of an adaptive adversary. We provide a formal definition of this primitive in Sect. 3.7.

3.5. *Garbled Circuits*

A core building block in our instance-dependent commitment schemes is garbled circuits [69]. 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 about the function but its output. Garbled circuits are an extremely useful tool for securely realizing any PPT function such that the input is distributed among an arbitrary number of players [14], and security holds in the presence of a static adversary. Toward introducing our definition of a garbled scheme, we denote vectors by bold lower-case letters and use the parameter n to denote the input and the parameter m to denote the output length for the Boolean circuit C.

Definition 3.9. (*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:

- $(\widetilde{\mathrm{C}}, \mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C}; r_{\mathsf{Grb}})$: is a probabilistic algorithm that takes as input a circuit C with n input wires and m output wires and returns a garbled circuit $\widetilde{\mathrm{C}}$, an encoding function ek and a decoding function dk.
- $\widetilde{\mathbf{x}} := \mathsf{Enc}(\mathsf{ek}, \mathbf{x})$ is a deterministic algorithm that takes an input a decoding function ek and an input $\mathbf{x}$ and returns an encoded input $\widetilde{\mathbf{x}}$. In this work, we consider *decomposable garbled schemes*. Namely, the algorithm takes multiple input bits $\mathbf{x} = (x_1, \dots, x_n)$, runs $\mathsf{Enc}(\mathsf{ek}, \cdot)$ on each x_i and returns the garbled inputs $\widetilde{x}_1$ through $\widetilde{x}_n$, denoted by input labels.[11]
- $\widetilde{\mathbf{y}} := \mathsf{Eval}(\widetilde{\mathrm{C}}, \widetilde{\mathbf{x}})$: is a deterministic algorithm that takes as input a garbled circuit $\widetilde{\mathrm{C}}$ and encoded input $\widetilde{\mathbf{x}}$ and returns encoded outputs $\widetilde{\mathbf{y}}$.
- $\mathbf{y} := \mathsf{Dec}(\mathsf{dk}, \widetilde{\mathbf{y}})$: is a deterministic algorithm that takes as input a decoding function dk and encoded output $\widetilde{\mathbf{y}}$ and returns a final output $\mathbf{y}$.

The security of garbled schemes follows by correctness and privacy stated below. In this work, we suggest to enhance the traditional view of this object with two additional algorithms that allow to capture the security properties we need for our commitment protocols. To be concrete, our notion of garbled circuits includes an additional algorithm for an oblivious generation of a garbled circuit. Namely, given the randomness used to produce a garbled circuit $\widetilde{\mathrm{C}}$ of some circuit C, this algorithm generates new randomness that explains $\widetilde{\mathrm{C}}$ as the outcome of the simulated algorithm. We stress 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. We further require that giving the encoding function ek and the garbled circuit $\widetilde{\mathrm{C}}$ it is possible to verify that $\widetilde{\mathrm{C}}$ was honestly generated by Grb. More formally,

1. **Perfect correctness** For all circuits $\mathrm{C} : \{0,1\}^n \mapsto \{0,1\}^m$ and for all $\mathbf{x} \in \{0,1\}^n$ it holds that,

$$\Pr[(\widetilde{\mathrm{C}}, \mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C}); \widetilde{\mathbf{x}} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathbf{x}); \widetilde{\mathbf{y}} \leftarrow \mathsf{Eval}(\widetilde{\mathrm{C}}, \widetilde{\mathbf{x}}); \mathbf{y} := \mathsf{Dec}(\mathsf{dk}, \widetilde{\mathbf{y}}) : \mathrm{C}(\mathbf{x}) = \mathbf{y}] = 1.$$

2. **Privacy** There exists a PPT algorithm SimGC such that for any polynomial-size circuit $\mathrm{C} : \{0,1\}^n \mapsto \{0,1\}^m$ and for all $\mathbf{x} \in \{0,1\}^n$ it holds that,

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

where $\mathbf{y} = \mathrm{C}(\mathbf{x})$, $(\widetilde{\mathrm{C}}, \mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C})$ and $\widetilde{\mathbf{x}} := \mathsf{Enc}(\mathsf{ek}, \mathbf{x})$.

3. **Oblivious sampling** There exists a PPT algorithm OGrb such that for any polynomial-time circuit $\mathrm{C} : \{0,1\}^n \mapsto \{0,1\}^m$ and for all input/output pairs $(\mathbf{x}, \mathbf{y})$ such that $\mathrm{C}(\mathbf{x}) = \mathbf{y}$ it holds that,

[11] Note that the notion of decomposability is similar to the notion of projective garbled schemes specified in [12].

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

where $(\widetilde{\mathrm{C}}, \mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C}; r_{\mathsf{Grb}})$.

4. **Verifiability** There exists a PPT algorithm Ver that takes an input garbled a circuit $\widetilde{\mathrm{C}}$ an encoding function ek and returns a bit b such that $b = 1$ only if there exists randomness r_{Grb} such that $(\widetilde{\mathrm{C}}, \mathsf{ek}, \cdot) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C}; r_{\mathsf{Grb}})$.

We now prove the all these properties are met by Yao's construction [61,69].

Theorem 3.10. *Assume the existence of one-way functions. Then, the notion of garbled circuits as introduced in* [61,69] *meets the above four requirements.*

Proof. We briefly demonstrate that every property is met.

1. **Perfect correctness** This property is achieved by employing the point-and-permute optimization [67] embedded within 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.
2. **Privacy** To support the next property of oblivious sampling, we slightly modify the simulation from [61] and require that the underlying symmetric key encryption has an additional property of oblivious ciphertext generation (where a ciphertext can be sampled without the knowledge of the plaintext). Then, our simulated garbling of a gate produces a garbled table using three obliviously generated ciphertexts and one ciphertext that encrypts the output label. Adapting the indistinguishability proof from [67] for our simulation is straightforward. We lastly note that our simulation can be realized based on one-way functions. Specifically, the classic symmetric key encryption scheme based on pseudorandom functions (PRFs) allows oblivious sampling of ciphertexts.[12]
3. **Oblivious sampling** Our simulation from above easily supports oblivious sampling of garbled circuits. Namely, switching from a real garbled circuit to a simulated one can be shown by explaining the three inactive ciphertexts for each gate (namely the ciphertext that remains non-decrypted during the evaluation phase) as obliviously sampled.
4. **Verifiability** Finally, this property holds with respect to existing garbling schemes, as the encoding information includes the pairs of secret keys that are associated with the input wires of the circuit and allows to recompute the entire garbling and verifying the correctness of each gate. □

3.6. *Randomized Encoding*

We review the definition of randomized encoding from [2,48]. The following definition is produced almost verbatim from [2].

[12]More formally, let $\mathrm{F} : \{0,1\}^\kappa \times \{0,1\}^\kappa \mapsto \{0,1\}^\kappa$ denote a PRF function. Then encrypting a message $m \in \{0,1\}^\kappa$ is carried out by sampling a random $r \leftarrow \{0,1\}^\kappa$ and returning $(\mathrm{F}_k(r) \oplus m, r)$. Furthermore, obliviously sampling a ciphertext is achieved by sampling two κ-bits strings. By the pseudorandomness of F, an obliviously generated ciphertext is indistinguishable from a real one.

Definition 3.11. (*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 over $n \in \mathbb{N}$:

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

In [6], Applebaum et al. introduced 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. This is formalized by requiring two functions $\widehat{f}_{\text{OFF}}$ and $\widehat{f}_{\text{ON}}$ where $\widehat{f}_{\text{OFF}}$ on input r outputs the offline encoding and $\widehat{f}_{\text{ON}}$ on input x and the same randomness r outputs the online encoding. Online efficiency here means that the complexity of $\widehat{f}_{\text{ON}}$ is smaller than the circuit size. For example, the standard garbling scheme meets this requirement. Specifically, the offline phase can be viewed as the garbled circuit, whereas the online phase, given an input x, are the keys corresponding the bits of x. Furthermore, the online complexity is proportional to the input size of the function alone.

In our construction, we are not concerned specifically with the online complexity, but require that the online encoding satisfy a "decomposable" property. Loosely speaking, decomposable randomized encoding requires that the online encoding can be split into encoding functions, one function corresponding to each input bit [3,29]. The online encoding is then the concatenation of the output of each of the encodings. The only thing shared between the encoding functions is the randomness. We will require a slightly different decomposable property. Namely, we need that the online encoding is a projection on the randomness. In other words, $\widehat{f}_{\text{ON}}(x,r)$ is defined by a family of linear mappings $\mathcal{P} = \{P_x(r)\}_{x\in\{0,1\}^n, r\in\{0,1\}^m}$ where $P_x(r)$ is a subset of $[m]$ that is defined by the string x, for $|r| = m$. Note that the standard notion of garbled circuits satisfies this property where the randomness r corresponds to a list of pairs of random strings $((r_1^0, r_1^1), \ldots, (r_n^0, r_n^1))$ and $P_x(r)$ is defined by the matrix that maps r into the vector $(r_1^{x_1}, \ldots, r_n^{x_1})$.

We also require the randomized encoding to be perfectly correct. We formalize these properties below:

1. **Perfect correctness** We say that $\widehat{f} = (\widehat{f}_{\text{OFF}}, \widehat{f}_{\text{ON}})$ is a *perfectly correct* randomized encoding of a function f, if for every x and r, it holds that:

$$B(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x,r)) = f(x).$$

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,

$$\{r, \mathcal{S}_{\mathrm{OFF}}(y, r), \mathcal{S}_{\mathrm{ON}}(\mathsf{st}, r)\}_{r \leftarrow \{0,1\}^*} \overset{\mathrm{c}}{\approx} \{\mathsf{ORE}(x, r), \widehat{f}_{\mathrm{OFF}}(r), \widehat{f}_{\mathrm{ON}}(x, r)\}_{r \leftarrow \{0,1\}^*}$$

where st is a state output by $\mathcal{S}_{\mathrm{OFF}}$. Note that we consider the most general form of a simulator that is comprised from two distinct sub-algorithms $\mathcal{S}_{\mathrm{OFF}}$ and $\mathcal{S}_{\mathrm{ON}}$.

3. **Affine-projective form and robustness** We say that $\widehat{f} = (\widehat{f}_{\mathrm{OFF}}, \widehat{f}_{\mathrm{ON}})$ is in *affine-projective* form if there exists a family of affine projections $\mathcal{P} : \{0, 1\}^m \mapsto \{0, 1\}^m$ such that $\widehat{f}_{\mathrm{ON}}(x, r) = (v_r \oplus x, P_x(r))$ where vector v_r only depends on r and $P_x(r)$ is an affine projection function picked from a family $\mathcal{P}$.
A randomized encoding $\widehat{f} = (\widehat{f}_{\mathrm{OFF}}, \widehat{f}_{\mathrm{ON}})$ in affine projective form is a *robust encoding* of f if it holds that for every string r^* there exists no projection $P_x^* \in \mathcal{P}$ such that

$$B(\widehat{f}_{\mathrm{OFF}}(r^*), (v_{r^*} \oplus x, P_x^*(r^*))) \notin \{f(x), \perp\}.$$

We conclude with the following theorem, proving in Sect. 5.

Theorem 3.12. *Assume 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* $(\widehat{f}_{\mathrm{OFF}}, \widehat{f}_{\mathrm{ON}}, \mathcal{S}_{\mathrm{OFF}}, \mathcal{S}_{\mathrm{ON}})$ *with affine-projective property such that the offline complexity is* $O(s \cdot \mathsf{poly}(\kappa))$ *and the online complexity is* $O(n \cdot \mathsf{poly}(\kappa))$ *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 Sect. 5, we show how to realize such a randomized encoding based on any two-party secure computation protocol (that meets certain requirements), which in particular, is satisfied by the [69] and [40] 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 $O(s\kappa)$.

3.7. *Secure Two-Party Computation*

In the following, we present the notion of two-party adaptive security [15].

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 adversary's 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 post-execution 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 $\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x_0, x_1, z, \boldsymbol{r})$ 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 $\boldsymbol{r} = (r_{\mathcal{Z}}, r_{\mathcal{A}}, r_0, r_1)$. Let $\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x_0, x_1, z)$ denote a random variable describing $\mathbf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x_0, x_1, z, \boldsymbol{r})$ 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 $\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(\kappa, x_0, x_1, z, \boldsymbol{r})$ 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 $\boldsymbol{r} = (r_{\mathcal{Z}}, r_{\mathcal{S}})$. Let $\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(\kappa, x_0, x_1, z)$ denote a random variable describing $\mathbf{IDEAL}_{f,\mathcal{S},\mathcal{Z}}(\kappa, x_0, x_1, z, \boldsymbol{r})$ 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 3.13. 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:

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

Adaptive zero-knowledge As explained in [63], 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 post-execution 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:

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

Definition 3.14. 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 PPTreal-life adversary $\mathcal{A}$ and any PPT environment $\mathcal{Z}$, there exists a probabilistic PPT adaptive ideal-process adversary $\mathcal{S}$, such that

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

4. Warmup: Static Zero-Knowledge Proofs from 2PC

As a starting point, we demonstrate that our technique implies static ZK proofs from any two-party protocol that provides perfect correctness. Intuitively speaking, consider a two-party protocol that is secure 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 remark that this protocol can be made negligibly sound by standard sequential repetition [39] or by relying on parallel repetition along with statistically hiding commitments [28,46]. While this does give a simple zero-knowledge proof (that could be implemented with off-the-shelf garbled circuits implementations), the asymptotic complexity is worse than [50]. The strength of our construction will be explored in Sect. 6.1, where we extend this basic protocol to additionally obtain the input-delayed property.

More formally, let $f : \{0, 1\}^n \to \{0, 1\}$ be an arbitrary polynomial-time computable function. We define $g(a_1, a_2) = f(a_1 \oplus a_2)$ and view g as a two-party functionality. Then, let ρ^{OT}_g be any two-party protocol in the OT-hybrid model that realizes g with static semi-honest security. We will consider a slight variant of this protocol, denoted by Π^{OT}_g, where for every OT call we make the following modification: Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an IND-CPA symmetric key encryption scheme based on pseudorandom functions.

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.

Fig. 2. Static zero-knowledge proof for any language $\mathcal{L} \in \mathsf{NP}$.

- For every OT call where P_1's input is (s_0, s_1) and P_2's input is b, we require P_1 to send $(c_0 = \mathsf{Enc}_{k_0}(s_0), c_1 = \mathsf{Enc}_{k_1}(s_1))$ to P_2 and use random keys (k_0, k_1) as its input to the OT. Upon receiving k_b from OT functionality, P_2 will obtain s_b by decrypting c_b with k_b.

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. Our protocol is specified in Fig. 2.

Theorem 4.1. *Assume the existence of one-way functions. Then, the protocol presented in Fig. 2 is a static honest-verifier zero-knowledge proof for any language in NP.*

As this protocol only serves as a warmup toward presenting our main (and more involved) results, we only provide a proof sketch here and postpone the proof details to the next sections.

Proof sketch Completeness follows easily from the fact that the honest prover knows the witness ω, and thus, it can answer both challenges of the verifier. On the other hand, from the perfect correctness of Π_g^{OT}, a malicious prover cannot provide randomness and input for both parties that are consistent with τ since that would 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, a simulator can be defined by guessing the random challenge b and invoking the simulator for Π_g^{OT} for the case that P_{1+b} is corrupted. Indistinguishability of the real and simulated proofs follows from the privacy of Π_g^{OT} where simulated and real transcripts are indistinguishable in the presence of a single static corruption. $\square$

Finally, we note that our protocol implies the first black-box transformation from [69] and (the two-party variant of) [40] to *static zero-knowledge proof.*

5. Randomized Encoding from 2PC

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 in contrast to prior works that have established the usefulness of randomized encoding in constructing efficient multiparty computation [2,26,48].

Recalling the definition of protocol Π_g^{OT} from the prior section, we require that it satisfies 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 [43]) 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. Furthermore, 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.

Both the classic Yao's garbled circuit construction [69] and the [40] protocol satisfy these conditions in the OT-hybrid setting. In Sects. 5.1 and 5.2, we discuss these two realizations in more details.

Our randomized encoding We now proceed with the description of our robust randomized encoding of f with affine-projective property as formalized in Sect. 3.6 by specifying the functions $\widehat{f}_{\mathrm{OFF}}$, $\widehat{f}_{\mathrm{ON}}$ and the simulator $\mathcal{S}$. Toward describing our algorithms, we consider a real-world experiment carried out between parties P_1 and P_2 that engage in an execution of Π with environment $\mathcal{Z}$. Let $\mathbf{REAL}^{\mathrm{OT}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x, \boldsymbol{r})$ 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 Π in the OT-hybrid where the inputs are determined by $\mathcal{Z}$ and $\boldsymbol{r} = (r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, r_2)$. Let $\mathbf{REAL}^{\mathrm{OT}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x)$ denote a random variable describing $\mathbf{REAL}^{\mathrm{OT}}_{\Pi,\mathcal{A},\mathcal{Z}}(\kappa, x, \boldsymbol{r})$ where the random tapes are chosen uniformly. We denote by $\mathbf{IDEAL}_{g,\mathcal{S},\mathcal{Z}}(\kappa, x, \boldsymbol{r})$ 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 $\boldsymbol{r} = (r_{\mathcal{Z}}, r_{\mathcal{S}})$. Let $\mathbf{IDEAL}_{g,\mathcal{S},\mathcal{Z}}(\kappa, x)$ denote a random variable describing $\mathbf{IDEAL}_{g,\mathcal{S},\mathcal{Z}}(\kappa, x, \boldsymbol{r})$ 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. From the guarantees of the protocol Π_g^{OT}, we know there exists a simulator corresponding to adversary $\mathcal{A}_1$. Let this simulator be $\mathcal{S}_1$.

1. $\widehat{f}_{\mathrm{OFF}}(r)$: Let $r = (r_{\mathcal{S}_1})$. The offline encoding is obtained by $\mathcal{S}_1$ with randomness $r_{\mathcal{S}_1}$ until it sends the first message to the environment. Recall that $\mathcal{A}_1$ first statically corrupts P_1 and after completing the execution using Π sends the transcript of the messages to the environment. We define the output of $\widehat{f}_{\mathrm{OFF}}(r)$ to be the output of $\mathcal{S}_1$ where the input a_1 provided for party P_1 is sampled uniformly at random. We remark here that the particular environment $\mathcal{Z}$ that we will rely on in the security proof will sample inputs a_1 and a_2 uniformly at random subject to $a_1 \oplus a_2 = x$. The offline encoding will, however, not depend on x as it only requires a_1 to be distributed correctly (which is uniform).
2. $\widehat{f}_{\mathrm{ON}}(x, r)$: To obtain the online part, we continue the execution of $\mathcal{S}_1$ in the execution corresponding to the transcript τ generated by $\widehat{f}_{\mathrm{OFF}}(r)$. Recall that after sending τ, $\mathcal{A}_1$ adaptively corrupts P_2 and sends the input and random tape of P_2 to the environment. $\widehat{f}_{\mathrm{ON}}(x, r)$ 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 $\widehat{f}_{\mathrm{ON}}(x, r)$ is set to this view. The view contains (a_2, r_2, m_2) where r_2 is the random tape of P_2 output by $\mathcal{S}_1$ and m_2 is the communication received from the OT functionality.

Decoder The decoder B on input $(z_{\mathrm{OFF}}, z_{\mathrm{ON}})$ 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 $(\tau, (a_2, r_2, m_2))$ 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 and m_2 is the communication received from the OT functionality. Let $\mathcal{S}_2$ be the corresponding simulator. Then, the simulation algorithm of the randomized encoding $\mathcal{S} = (\mathcal{S}_{\mathrm{OFF}}, \mathcal{S}_{\mathrm{ON}})$ is defined as follows. Upon receiving $y = f(x)$, $\mathcal{S}_{\mathrm{OFF}}$ 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 $(\tau, (a_2, r_2, m_2))$ at the end of the execution. Then, the output of $\mathcal{S}_{\mathrm{OFF}}$ is τ and the output of $\mathcal{S}_{\mathrm{ON}}$ is (a_2, r_2, m_2).

Theorem 5.1. *Assume the existence of one-way functions and 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. We obtain an encoding with offline complexity $C_\Pi \kappa$ and online complexity $|x| + r_\Pi + \rho_\Pi$ where C_Π is the communication complexity of Π_g^{OT} in the OT-hybrid, ρ_Π is the number of bits received by P_2 from the OT functionality, r_Π is the randomness complexity of P_2 in Π_g^{OT}.*

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. Toward 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 $(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r))$ can be rewritten as $\mathbf{IDEAL}_{g,\mathcal{S}_1,\mathcal{Z}^*}(\kappa, x)$. 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 random variable $\mathbf{IDEAL}_{g,\mathcal{S}_2,\mathcal{Z}}(\kappa, x)$. More precisely,

$$(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r)) \equiv \mathbf{IDEAL}_{g,\mathcal{S}_1,\mathcal{Z}^*}(\kappa, x), \text{ whereas} \quad (1)$$
$$\mathcal{S}(f(x)) \equiv \mathbf{IDEAL}_{g,\mathcal{S}_2,\mathcal{Z}^*}(\kappa, x). \quad (2)$$

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:

$$\mathbf{IDEAL}_{g,\mathcal{S}_1,\mathcal{Z}^*}(\kappa, x) \overset{c}{\approx} \mathbf{REAL}^{\text{OT}}_{\Pi,\mathcal{A}_1,\mathcal{Z}^*}(\kappa, x), \text{ and} \quad (3)$$
$$\mathbf{IDEAL}_{g,\mathcal{S}_2,\mathcal{Z}^*}(\kappa, x) \overset{c}{\approx} \mathbf{REAL}^{\text{OT}}_{\Pi,\mathcal{A}_2,\mathcal{Z}^*}(\kappa, x). \quad (4)$$

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 have that

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

and the proof of indistinguishability of our simulation by $\mathcal{S}$ for the randomized encoding follows using a standard hybrid argument.

(Perfect) 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. From Eqs. 1 and 3, we have that $(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r)) \overset{c}{\approx}$ $\mathbf{REAL}_{\widetilde{\Pi},\mathcal{A}_1,\mathcal{Z}^*}(\kappa, x)$. Since P_2 outputs $f(x)$ except with negligible probability in the real-world experiment, it follows that $B(\widehat{f}_{\text{OFF}}(r), \widehat{f}_{\text{ON}}(x, r)) = f(x)$ except with negligible probability. If Π^{OT}_g is perfectly correct and the simulation by $\mathcal{S}_1$ is perfect, then our randomized encoding will be perfectly correct.

Complexity Finally, we measure the complexity of our encoding. The offline computational complexity is the computational complexity of P_1, and the online computational complexity is proportional to the computational complexity of P_2. The length of the offline encoding is the communication complexity of Π^{OT}_g, denoted by C_Π. The length of the online encoding is equal to the sum of P_2's input length $|a_2|$, its randomness $|r_2|$ and the communication between the OT and P_2 which is denoted by ρ_Π. That equals $|x| + r_\Pi + \rho_\Pi$. □

Next, we instantiate our randomized encoding using the Yao's garbling scheme based 2PC and the GMW protocol. We will additionally show that each of these instantiations are projective, perfectly correct and admits oblivious sampling.

5.1. *Robust RE from Garbled Circuits*

We next demonstrate that our construction can be realized based on Yao's garbling circuit-based 2PC. This is shown by proving that Yao's protocol from [61] achieves all the security requirements.

We do this in two parts. First, we show that the 2PC protocol satisfies semi-adaptivity and oblivious sampleability. Then, we show that the resulting randomized encoding achieves the required properties of perfect correctness, projective and oblivious sampleability. We will make a few modifications to the standard garbled circuit construction. We will require that the underlying encryption scheme satisfy two properties: (1) zero decryption error, and (2) pseudorandom ciphertexts. The standard IND-CPA encryption scheme based on pseudorandom functions satisfy these two properties, where the former is achieved when the garbling scheme is embedded with the point-and-permute optimization [67]. We will also modify the standard Lindell Pinkas simulation of the garbled circuit [61]; formally described under oblivious sampling below.

Semi-adaptivity Recall that semi-adaptivity requires that the protocol is secure in the presence of static corruption of P_1 followed by an adaptive (post-execution) corruption of P_2 by a semi-honest adversary. In the OT-hybrid, we can generate P_1's communication honestly using its input. Recall that in the standard Yao's protocol the communication contains the garbled circuit and the garbler's keys. Moreover, in our randomized encoding the P_1's input to the OT as the sender is not directly incorporated in the transcript.[13] The evaluator's (P_2's) keys are delivered using OT and are not present in the communication channel. At the end of the execution when P_2 is corrupted and its input is received, the simulator needs to generate (a_2, r_2, m_2) where a_2 is provided to the simulator. For the garbled circuit construction r_2 is the all 0's string and m_2 is the keys corresponding to the bits in a_2. As P_1 is honestly simulated, the keys to both values are known and hence m_2 can be obtained. Furthermore, the simulation is perfect.

Oblivious sampling We recall that oblivious sampling requires that the view of P_2, output by $\mathcal{S}_1$, can be explained as an output of $\mathcal{S}_2$. In other words, it must be shown that the adaptive simulation of (post-execution corrupted) P_2 can be explained as if P_2 was statically corrupted. To prove the latter, we must modify the way Lindell and Pinkas designed 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 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

[13]However, with our modification, indirectly the encrypted values of the sender's real inputs are in the transcript.

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 pseudorandom ciphertexts. Note that now it is possible to explain an adaptive simulation of P_2's view as a static one by relying on the pseudorandomness property of the underlying encryption scheme. Specifically, the simulator for the case that P_2 is adaptively corrupted will generate the garbled circuit honestly. Next, whenever needed to explain P_2's view as being statically corrupted, the garbled circuit can be explained as being obliviously sampled. That is, a single ciphertext within each gate will correspond to the "valid" ciphertext. The remaining three ciphertexts will be considered as being obliviously sampled.

Perfect correctness For the randomized encoding to be perfectly correct, we need the two-party protocol to be perfectly correct and the simulation of adversary $\mathcal{A}_1$ perfect. We argued above under semi-adaptivity that the simulation is perfect. If the underlying encryption scheme has zero decryption error, then the garbled circuit construction will be perfectly correct against semi-honest corruptions.

Affine-projective form and robustness The online encoding comprises the input a_2 to party P_2 and m_2 which is the communication between the OT functionality and P_2. a_2 is obtained by XORing x with a_1 where a_1 only depends on the randomness r used in the offline encoding and therefore satisfies the affine part. If $((k_1^0, k_1^1), \ldots, (k_n^0, k_n^1))$ are the keys (which are randomly sampled) used to encode the OT, then m_2 is a projection on these bits, namely $(k_1^{a_2^1}, \ldots, k_n^{a_2^n})$ where $a_2 = a_2^1 \cdots a_2^n$. The family of projections $\mathcal{P}$ that has efficiently checkable range can be described as any projection that chooses one of each of the two strings (k_i^0, k_i^1). The robustness of this scheme will follow from the fact that given an offline encoding with randomness r^*, there is no online encoding where the projection is restricted to the above family $\mathcal{P}$, that can result in an incorrect answer. This is because once the input keys are fixed and there is a zero decryption error, the evaluation by B is the evaluation of the garbled circuit which is deterministic. Namely, there is only one "active" path that leads to a correct output.

Realizing robust randomized encoding secure against adaptive choice of inputs based on [45] The work of [45] modifies the garbled circuit-based construction to achieve security against adaptive chosen inputs using the following approach: The offline encoding of [45] is an encryption of the garbled circuit under a special kind of encryption scheme where the key is revealed in the online phase. The special encryption scheme allows the encryptor to equivocate the ciphertext to M different possible plaintexts (with certain restrictions) where M is a parameter chosen for the security proof.[14] We remark that the honest encoding and the simulated encoding will not use this equivocation property, and reveal only one specific key in the onling phase. However, the proof of security involves a sequence of hybrid steps that will make use of the different keys.

We employ this construction in the scheme described above with a slight modification to obtain a robust randomized encoding that is secure against adaptive choice of inputs. We modify the two-party protocol that is used in the transformations. The

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

garbler transmits the encrypted garbled circuit and transmits the key under which it is encrypted using a 1-out-of-M oblivious transfer. The honest P_1 uses only one key k and sets all strings in the 1-out-of-M OT to be the key k. P_2, on the other hand, acting as the receiver, selects a random index from 1 to M. The semi-adaptive simulation by $\mathcal{S}_1$ of such a protocol will be same as before, as it will simply follow P_1's strategy honestly and then later when P_2 is corrupted, the simulator selects a random index for P_2 when simulating P_2's view. The simulation of a static corruption by $\mathcal{A}_2$ works by having the simulated key set as all the sender's strings in the 1-out-of-M oblivious transfer. The proof of correctness of the simulation by $\mathcal{S}_2$, however, will rely on the proof of [45]. The affine-projective property will follow essentially as before since we modeled the key transfer as an oblivious transfer.

5.2. *Robust RE from the GMW Protocol*

We begin by recalling the basic protocol from [40]. The parties P_1 and P_2 first create XOR shares of their respective inputs $x = x_1 \oplus x_2$ and $y = y_1 \oplus y_2$, 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 inputs. 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 share, while P_1 uses s. Finally, P_1 transmits its shares 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.

Semi-adaptivity We demonstrate that the GMW protocol is semi-adaptive. In fact, in the OT-hybrid the protocol is even fully adaptive. Nevertheless, we explicitly provide the semi-adaptive simulator. Upon statically corrupting $\mathcal{A}_1$, the simulator will generate P_1's communication honestly using P_1's input. Once $\mathcal{S}_1$ learns the input a_2 of P_2, it generates P_2's view (a_2, r_2, m_2) by simply running P_2's code honestly with the exception that the XOR shares for the input are fixed (as one of the shares appears in the transcript), and that will determine r_2. m_2, which is the communication between the OT functionality and P_2, can be determined by running the code of P_2 honestly and using the inputs fed by P_1 to the OT functionality. It follows just as in our garbled circuit construction that the simulation by $\mathcal{S}_1$ is perfect.

Oblivious sampling We need to show that the view of P_2 output by $\mathcal{S}_1$ can be explained as an output of $\mathcal{S}_2$. First we recall the standard simulation of adversary $\mathcal{A}_2$ that statically corrupts P_2. The simulator obtains P_2's input a_2. It computes a random XOR share of a_2 into two shares and sends one share to P_1. Then, it simulates each of the OT calls so that P_2 receives a random bit (this is because the actual result of each multiplication is masked with an independent random bit s). If y is the share of the output bit computed by P_2, it feeds $f(x) \oplus y$ as the message received from P_1. Recall that we make a slightly modification in our protocol (see Sect. 4), where the actual inputs of the sender for each OT are encrypted under a random key, and the corresponding keys are used as inputs to the OT functionality. To simulate the encryptions sent, it suffices to encrypt the one value that the receiver will decrypt and simulate the rest of the three encryptions by supplying random strings (recall that our encryption has pseudorandom ciphertexts). Given this simulation description by $\mathcal{S}_2$, we can explain the view of P_2 output of $\mathcal{S}_1$ as follows: It is easy to see that the input, shares of the input and shares of the output can be demonstrated as generated by $\mathcal{S}_2$ given the input a_2 and output $f(x)$. The only non-trivial part is demonstrating the communication for the OT generated by $\mathcal{S}_1$ was actually generated by $\mathcal{S}_2$. This can be done by showing that except for the one out of the four ciphertexts decrypted by P_2 in each OT, the remaining three ciphertexts were simply random strings.

Perfect correctness For the randomized encoding to be perfectly correct, we need the two-party protocol to be perfectly correct and the simulation of adversary $\mathcal{A}_1$ perfect. Recall that the standard GMW is perfectly correct. It further remains perfectly correct even with our modification to OT calls as long as there is zero decryption error (see Sect. 4). We argued above under semi-adaptivity that the simulation is perfect. Therefore, if the underlying encryption scheme has zero decryption error, then we have perfect correctness against semi-honest corruptions.

Affine-projective form and robustness The online encoding comprises the input a_2 to party P_2 and m_2 which is the communication between the OT functionality and P_2. a_2 is obtained by XORing x with a_1 where a_1 only depends on the randomness r used in the offline encoding. More formally, if $a_2 = x \oplus a_1$ and a_1 is a substring of r. Therefore, it satisfies the affine property. For every OT call, if (k_0, k_1) are the keys (which are randomly sampled) used as inputs to the oblivious-transfer, then m_2 contains k_b where b is P_2's input. As k_0, k_1 are substrings of r and k_b is a projection, the randomized encoding satisfies the projective property. To argue robustness, first we observe that the keys for every OT call are bound to the randomness, where the actual sender values to the OT by our modification are in the transcript, encrypted with these keys. Therefore, given r that is valid for the offline encoding and a corresponding online encoding, we obtain a valid transcript of an execution between P_1 and P_2. Robustness then follows from the perfect correctness of the underlying GMW protocol and zero decryption error of the underlying CPA encryption scheme.

The efficiency of our randomized encoding As shown above, both garbled schemes [61, 69] and the [40] protocol satisfy the required properties to realize our randomized encoding. Thus, if we rely on the former protocol, the offline complexity is $O(s \cdot \mathsf{poly}(\kappa))$, whereas the online complexity is $n \cdot \mathsf{poly}(\kappa)$, 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 [40], we get that the online and offline complexities are both $O(s \cdot \mathsf{poly}(\kappa))$. Finally, our robust randomized encoding secure against adaptive input based on [45] has an offline efficiency of $O(s \cdot \mathsf{poly}(\kappa))$ and an online efficiency of $O((d + n) \cdot \mathsf{poly}(\kappa))$ where d is width of the circuit implementing the computed function.

6. Input-Delayed Proofs

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

6.1. *Input-Delayed Zero-Knowledge Proofs*

In [62], Lapidot and Shamir provided a 3-round witness-indistinguishable (WI) proof of knowledge for Graph Hamiltonicity with a special "input-delayed" property: Namely, the prover uses the statement to be proven only in the last round. Recently, in [22] 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 to use randomized encoding that is secure against adaptive chosen inputs, to realize input-delayed zero-knowledge proofs. Then, relying on the recent construction of such a randomized encoding [45] 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. Roughly speaking, the input-delayed property allows an honest prover to generate all the messages except the last one without the knowledge of the statement. Consequently, the soundness and zero-knowledge property have to incorporate the possibility of the statement being adversarially chosen. Intuitively, soundness is required to hold even if the cheating prover adaptively chooses the statement before the last round. Zero-knowledge, on the other hand, is required to hold even if the malicious verifier chooses the statement before the last round. We next formalize this notion:

Definition 6.1. (*Input-delayed special-sound zero-knowledge proof*) A k-round protocol $(\mathcal{P}, \mathcal{V})$ for an NP language $\mathcal{L}$ with NP-relation $\mathcal{R}$ is an input-delayed zero-knowledge proof if it is complete and the following properties hold:

Adaptive special soundness There exists a polynomial-time extractor algorithm X, such that for every polynomial-time machine P^*, there exists a negligible function $\nu(\cdot)$ such that the following holds for every auxiliary input z for P^*. Let τ_1, τ_2 denote two transcripts between $P^*(1^n, z)$ and $V(1^n)$ where V uses the same randomness for the first $k-2$ rounds (i.e., the first $k-2$ rounds are identical in both transcripts). Then, the probability over the randomness is used to generate the transcripts that:

1. τ_1 and τ_2 are accepting transcripts with statements x_1 and x_2, respectively, and different messages only within the $k-1$ round, and
2. $X(\tau_1, \tau_2)$ does not output (ω_1, ω_2) such that $\omega_1 \in R_L(x_1), \omega_2 \in R_L(x_2)$,

is smaller than $\nu(n)$.

Input-delayed zero-knowledge For every pair of PPT algorithms $\mathcal{V}_1^*, \mathcal{V}_2^*$, there exists a pair of PPT algorithms (S_1, S_2), such that following two distributions are indistinguishable.

- $\{\mathbf{REAL}_{\mathcal{V}_1^*,\mathcal{V}_2^*}(1^n, z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$, and
- $\{\mathbf{IDEAL}^{S_1,S_2}_{\mathcal{V}_1^*,\mathcal{V}_2^*}(1^n, z)\}_{n\in\mathbb{N},z\in\{0,1\}^*}$.

where the real experiment $\mathbf{REAL}_{\mathcal{V}_1^*,\mathcal{V}_2^*}(1^n, z)$ proceeds as follows: The honest prover P on input 1^n interacts with $\mathcal{V}_1^*(1^n, z)$ for $k-1$ rounds. The view of $\mathcal{V}_1^*$ is fed as input to $\mathcal{V}_2^*$. $\mathcal{V}_2^*$ first generates (x, ω) and sends it to P in a special input tape and continues to interact with P to receive the kth-round message. The output of the experiment is the output of $\mathcal{V}_2^*$ if $\omega \in R_L(x)$ and $\perp$ otherwise.

The ideal experiment $\mathbf{IDEAL}^{S_1,S_2}_{\mathcal{V}_1^*,\mathcal{V}_2^*}(1^n, z)$ proceeds as follows: The simulator S_1 on input $(1^n, z)$ outputs a view view and state st. $\mathcal{V}_2^*$ on input view outputs (x, ω). The output of the experiment is the output of S_2 on input $(1^n, st, x)$ if $\omega \in R_L(x)$ and $\perp$ otherwise.

Lemma 6.1. *Assume the existence of robust randomized encoding with the affine projective property that is secure against an adaptive chosen input, and one-way functions. Then, there exists 4-round input-delayed special-sound zero-knowledge proof according to Definition 6.1 for any language in* NP.

Proof. Given an NP-relation $\mathcal{R}$ and a constant C, define the function f as

$$f[C](x, D) = (\mathcal{R}(x, C \oplus D), x, D).$$

Let com be the commitment scheme based on one-way functions [64]. Then, the protocol proceeds as follows:

1. $\mathcal{V} \to \mathcal{P}$: $\mathcal{V}$ sends the first message η for the commitment scheme com.
2. $\mathcal{P} \to \mathcal{V}$: $\mathcal{P}$ samples randomness r^0, r^1 and random strings ω_0^0, ω_0^1. Let $(\widehat{f}^{\tau}_{\text{OFF}}, \widehat{f}^{\tau}_{\text{ON}}, S^{\tau}_{\text{OFF}}, S^{\tau}_{\text{ON}})$ be a robust randomized encoding of the function $f[\omega_0^{\tau}](\cdot,\cdot)$. $\mathcal{P}$ sends $(F^0_{\text{OFF}}, \sigma^0)$ and $(F^1_{\text{OFF}}, \sigma^1)$ to $\mathcal{V}$ where $F^{\tau}_{\text{OFF}} = \widehat{f}^{\tau}_{\text{OFF}}(r^{\tau})$, $\sigma^{\tau} = \mathsf{com}_{\eta}(r^{\tau}, \omega_0^{\tau})$ for $\tau \in \{0, 1\}$ and the commitment is made bit-by-bit.
3. $\mathcal{V} \to \mathcal{P}$: The verifier sends a random challenge $b \leftarrow \{0, 1\}$.
4. $\mathcal{P} \to \mathcal{V}$: Upon receiving the input statement x and witness ω, the prover sends:
   (a) Decommitment to every bit in σ^b, and
   (b) $F^{1-b}_{\text{ON}} = \widehat{f}^{1-b}_{\text{ON}}(\mathsf{inp}^{1-b}, r^{1-b})$ along with decommitments of specific bits of r^{1-b} from σ^{1-b} determined by the positions induced by the projection $P_{\mathsf{inp}^{1-b}}(r^{1-b})$ where $\mathsf{inp}^{1-b} = (x, \omega \oplus \omega_0^{1-b})$.

   The verifier accepts only if:
   (a) $F^b_{\text{OFF}} = \widehat{f}^b_{\text{OFF}}(r^b)$ is a valid offline encoding of the function $f[\omega^*](\cdot,\cdot)$ where σ^b was decommitted to (r^b, ω^*).
   (b) $(F^{1-b}_{\text{OFF}}, F^{1-b}_{\text{ON}})$ decodes to $(1, x, \cdot)$.
   (c) Let F^{1-b}_{ON} be of the form $(\cdot, \Delta)$, then the specific bits decommitted to in σ^{1-b} are exactly Δ.

Completeness follows directly from the correctness of the underlying randomized encoding scheme. Adaptive special soundness, on the other hand, follows from the robustness of the randomized encoding. If a prover manages to convince the verifier for two different second messages, namely challenge 0 and challenge 1 with possibly different (adaptively chosen) statements, then we need to demonstrate that witnesses can be extracted for both the statements. More precisely, let τ_1 and τ_2 be two transcripts with different challenges but same two messages. We define an extractor X that on input τ_1, τ_2 that proceeds as follows: As both transcripts are convincing, there must be $r^0, \omega_0^0, \omega_1^0, r^1, \omega_0^1, \omega_1^1$ such that

- $F_{\mathrm{OFF}}^0 = \widehat{f}_{\mathrm{OFF}}^0(r^0)$ and $(F_{\mathrm{OFF}}^0, F_{\mathrm{ON}}^0)$ decodes to $(1, x_0, \omega_1^0)$.
- $F_{\mathrm{OFF}}^1 = \widehat{f}_{\mathrm{OFF}}^1(r^1)$ and $(F_{\mathrm{OFF}}^1, F_{\mathrm{ON}}^1)$ decodes to $(1, x_1, \omega_1^1)$.

X outputs $(\omega_0^0 \oplus \omega_1^0, \omega_0^1 \oplus \omega_1^1)$. The correctness of the extractor will follow from the robustness of the underlying randomized encoding scheme. Recall that the robustness property for our randomized encoding holds if there is a valid r corresponding to the offline part such that the second part of the online encoding exactly contains the projection of this r. As the bits of r are committed to in our protocol, the projection is enforced by requiring that the corresponding bits of r to be decommitted, we have that by the binding property of the commitment, robustness holds with high probability. Therefore, robustness holds and the probability with which the prover can provide an online part that will decode to a wrong value is negligible. Since the decoded output is correct with high probability, it follows that the extractor will succeed with the same probability.

Finally, we demonstrate the zero-knowledge property. Here, the simulator guesses the challenge of the verifier in advance and computes the first message according to this challenge following the honest commitment algorithm. In more detail, given $\mathcal{V}_1^*$ and $\mathcal{V}_2^*$, we define a pair of (oracle) algorithms S_1, S_2. S_1 with oracle access to $\mathcal{V}_1^*$ proceeds as follows:

1. S_1 obtains η from the verifier.
2. S_1 makes a guess for the verifier's challenge b. It next generates a first message to be fed to $\mathcal{V}_1^*$. It samples randomness r^b, r_{sim} and random strings ω_0^b and feeds $(F_{\mathrm{OFF}}^0, \sigma^0)$ and $(F_{\mathrm{OFF}}^1, \sigma^1)$ internally to $\mathcal{V}_1^*$ where
   - $F_{\mathrm{OFF}}^b = \widehat{f}_{\mathrm{OFF}}^b(r^b)$ and $\sigma^b = \mathsf{com}_\eta(r^b, \omega_0^b)$, and
   - $F_{\mathrm{OFF}}^{1-b} = S_{\mathrm{OFF}}(r_{sim})$ and σ^{1-b} is sampled as a random string.
3. If $\mathcal{V}_1^*$ responds with challenge b, then S_1 simply outputs r^b, r_{sim}, ω_0^b and decommitments of σ^b as its state st and the view of $\mathcal{V}_1^*$ in the internal emulation as view. Otherwise, it rewinds to step 1 and makes a new guess for b.

$\mathcal{V}_2^*$ takes as input view and outputs (x, ω). S_2 takes an input $(1^n, st, x)$ and proceeds as follows:

- It samples ω_1^{1-b} and feeds $\mathcal{V}_2^*$ the decommitment of σ^b and $S_{\mathrm{ON}}^{1-b}((1, x, \omega_0^{1-b}), r_{sim})$ as the third message and outputs whatever $\mathcal{V}_2^*$ outputs.

Indistinguishability of the simulation follows directly from the indistinguishability of the simulation of the randomized encoding. □

The work of Hemenway et al. [45] shows how to obtain a randomized encoding secure against adaptively chosen inputs. We show in Sect. 5.1 how to extend it to achieve the stronger robustness property. Combining their work with our construction, we have the following corollary.

Corollary 6.2. *Assume the existence of one-way functions. Then, for any* NP*-relation* $\mathcal{R}$*, there exists a 4-round input-delayed special-sound ZK proof with communication complexity* $O(s \cdot \mathsf{poly}(k))$ *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.

Remark 6.2. Our protocol can be made 3-round if the underlying assumption is one-way permutation, removing the need of sending the first message of the verifier.

Remark 6.3. The zero-knowledge protocol described in this section is not an interactive proof in the traditional sense as it does not satisfy the standard soundness condition. Ideally, one would hope that no cheating prover can prove a false statement with a probability beyond a probability of $1/2$. However, in our protocol, as the prover can adaptively determine a statement for each challenge of the verifier, such a claim is meaningless as the prover need not fix the same statement for both the challenges. In the next section, we will compile this protocol to achieve the standard notion, where no cheating prover can convince a verifier of any false statement beyond a negligible probability.

6.2. *Commit-and-Prove Zero-Knowledge Proofs*

In the “commit-and-prove” paradigm, the prover first commits to its witness and then proves that the statement, along with the decommitment value maintains the underlying NP relation. This paradigm is useful for constructing maliciously secure protocols [20, 40]. In this section, we show how to design such an *input-delayed* proof, namely, where the statement is determined only at the last round and the underlying commitment scheme is used in a black-box way. Specifically, in this input-delaying flavor the witness is known ahead of time but not the statement, and hence not the NP relation.

As such we can modify the previous protocol to achieve this. Recall that in the first round, the prover commits to one of the two shares of the witness. We can additionally require the prover to commit to both shares. However, this does not completely solve the problem as we need to show that the witness used for both repetitions of the randomized encoding are the same. Namely, $\omega_0^0 \oplus \omega_1^0 = \omega_0^1 \oplus \omega_1^1$. Furthermore, the protocol will, at best, achieves soundness half (see Remark 6.3).

In order to improve the soundness parameter we need to repeat the basic proof sufficiently many times in parallel. This, however, does not immediately work as the dishonest prover may use shares of different messages for each proof instance. In order to overcome this problem we use the [51] approach to add a mechanism that verifies the consistency of the shares. Namely, suppose we wish to repeat the basic construction (from the previous section) in parallel $N = O(t)$ times where $t = O(\kappa)$ and κ is the security parameter. Unlike the basic protocol from the previous section, we describe our protocol in this section in a commitment hybrid model where both the prover and the verifier have access to an ideal commitment functionality $\mathcal{F}_{\text{COM}}$. Specifically, whenever we say $\mathsf{com}(x)$ in this protocol, we imply that the party sends x to the $\mathcal{F}_{\text{COM}}$ functionality. In Sect. 6.3 we discuss how to instantiate $\mathcal{F}_{\text{COM}}$.

- The verifier picks a random t-subset I of $[N]$ with repetitions. Namely, it chooses $i_1, \ldots, i_t$ uniformly at random from $[N]$. It also picks t random challenge bits $\{ch_i\}_{i \in I}$ and commits to them.
- The prover then continues as follows:
  1. It first generates N independent XOR sharings of the witness ω, say $\{\omega_{i,0}, \omega_{i,1}\}_{i \in [N]}$.
  2. Next, it generates the views of $2N$ parties $P_{i,0}$ and $P_{i,1}$ for $i \in [N]$ executing a t-robust t-private MPC protocol, where $P_{i,j}$ has input $\omega_{i,j}$, that realizes the functionality that checks if $\omega_{i,0} \oplus \omega_{i,1}$ are equal for all i. Let $V_{i,j}$ be the view of party $P_{i,j}$.
  3. Next, it computes N offline encodings of the following set of functions:

     $$f[\omega_{i,0}, V_{i,0}](x, \omega_{i,1}, V_{i,1}) = (b, x, \omega_{i,1}, V_{i,1}) \text{ with offline encoding } \widehat{f_i}^{\mathrm{OFF}}$$

     for $i \in [N]$, where $b = 1$ if and only if $\mathcal{R}(x, \omega_{i,0} \oplus \omega_{i,1})$ holds and the views $V_{i,0}$ and $V_{i,1}$ are consistent with each other.
  4. Finally, the prover sends:

     $$\left\{\widehat{f_i}^{\mathrm{OFF}}(r_i), \mathsf{com}(r_i), \mathsf{com}(\omega_{i,0}), \mathsf{com}(\omega_{i,1}), \mathsf{com}(V_{i,0}), \mathsf{com}(V_{i,1})\right\}_{i \in [N]}.$$

- The verifier decommits to all its challenges.
- For every index i in the t subset the prover replies as follows:
  - If $ch_i = 0$ then it decommits to r_i, $\omega_{i,0}$ and $V_{i,0}$. The verifier then checks if the offline part was constructed correctly (as in our basic proof).
  - If $ch_i = 1$ then it sends $\widehat{f_i}^{\mathrm{ON}}(r_i, x, \omega_{i,1}, V_{i,1})$ and decommits $\omega_{i,1}, V_{i,1}$ and specific bits of r_i determined by the projection in $\widehat{f_i}^{\mathrm{ON}}$. The verifier checks if the online encoding is consistent with the bits of r_i decommitted and then runs the decoder and checks if it obtains $(1, x, \omega_{i,1}, V_{i,1})$.

  Furthermore, from the decommitted views V_{i,ch_i} for every index i that the prover sends, the verifier checks if the MPC-in-the-head protocol was executed correctly and that all the views are consistent.

Theorem 6.3. *The above protocol is a commit-and-prove input-delayed zero-knowledge proof with negligible soundness for any language in* NP *in the* $\mathcal{F}_{\mathrm{COM}}$*-hybrid.*

Proof. We prove soundness of this protocol in two steps. First, we show along the same lines as [51] that except with negligible probability all shares reconstruct to the same value. Then it follows that for a false statement, the probability that an adversary can cheat in all t repetitions determined by the set I is at most 2^{-t}. The crucial idea is that we implement the MPC protocol from Step 2 using the [51] approach of MPC-in-the-head. More formally, we show that if the shares are inconsistent even in one of the repetitions, the prover will be caught with very high probability. We stress that we cannot rely on the proof presented in [51] because in the soundness proof the authors rely on the fact that every possible t-subset can be opened by the verifier. However, in our case, the verifier can only open restricted subsets of t views. Namely, for any index $i \in [N] \cap I$, it can choose to open either $V_{i,0}$ or $V_{i,1}$, but not both simultaneously. We therefore (re-)prove the soundness in our setting. Our proof starts the same way as [51], by considering an inconsistency graph G that has nodes $(i, 0)$ and $(i, 1)$ for $i \in [N]$ and an edge between two nodes (i, b_i) and (j, b_j) if the corresponding views V_{i,b_i} and V_{j,b_j} (that have been committed to in the first prover message), are inconsistent. Depending on the graph G, there are two cases:

Case 1: G **contains a vertex cover** B **of size at most** t: In this case, from the t-robustness of the MPC, we can conclude that if the verifier chooses any party outside the set of parties in B, the prover will be caught as these parties will output a value only in the support of the function and in the case of a false statement this can only be 0. Therefore, we simply estimate the probability that the verifier in our protocol opens the view of a party not in B. Since the size of B is at most t, and only one of each $(i, 0)$ and $(i, 1)$ can be opened, there will be at most t distinct values i such that (i, b_i) is in B for some value $b_i \in \{0, 1\}$. This means there are at least $N - t$ values for i such that neither $V_{i,0}$ or $V_{i,1}$ are in B and if such an $i \in I$, the prover is caught. Hence, the probability that the verifier misses all parties outside B is $(\frac{t}{N})^t$ which is negligible for $N = 4t$.

Case 2: G **has no vertex cover** B **of size less than** t: This means that there is a matching of size of at least $\frac{t}{2}$. In this case, we use the fact that the prover is caught if both vertices incident on an edge are opened by the verifier. We consider two sub-cases.

Subcase 1 Suppose the matching contains edges between $(i_j, 0)$ and $(i_j, 1)$ for $t/4$ distinct indices $i_1, \ldots, i_{t/4}$. Since the function $f_{\omega_{i,0}, V_{i,0}}$ checks the consistency of these pair of views $V_{i,0}$ and $V_{i,1}$, it must be the case that for every $j \in [t/4]$ such that $i_j \in I$, the prover is caught for one of the two values ch_{i_j} can take. We show that this happens except with negligible probability. First, we observe that the probability of I containing at most $t/12$ indices from $\{i_1, \ldots, i_{t/4}\}$ is $\left(\frac{N-t/4}{N}\right)^{t-t/12} < (1 - \frac{1}{16})^{11t/12}$ is negligible. Then, conditioned on having at least $t/12$ indices from $\{i_1, \ldots, i_{t/4}\}$ in I, the probability that none of the ch_i values for $i \in I \cap \{i_1, \ldots, i_{t/4}\}$ is the value that makes the prover caught is at most $(\frac{1}{2})^{t/12}$. Therefore, by a union bound, except with negligible probability, the prover is caught in this subcase.

Subcase 2 Suppose that there are fewer than $t/4$ such edges in the matching. Then, there are at least $t/4$ pairs of indices (i, j) such that there is an edge between (i, b_i) and (j, b_j) such that $i \neq j$, for some value b_i, b_j. Call this set of edges E. We now select edges using the following strategy: Pick a random edge from E, say between (i, b_i) and (j, b_j) and add it to the set E^*. Now remove edges from E that are incident on vertices with index i or j. Repeat until there are no more edges. Since every index i can be involved in at most two edges in a matching (namely one edge with vertex $(i, 0)$ and the other $(i, 1)$) for every edge added to E^* we remove at most two edges from the matching. Since E is of size at least $t/4$. the size of E^* is at least $t/12$. Let's denote the vertices incident on the edges in E^* by $\{(i_j, b_{i_j}), (\tilde{i}_j, b_{\tilde{i}_j})\}$ where $j \in \{1, \ldots, t/12\}$. By our selection strategy we have that all indices in $\{i_1, \tilde{i}_1, \ldots, i_{t^*}, \tilde{i}_{t^*}\}$ are distinct.

Next, we show that with high probability the views opened by the verifier correspond to at least one edge in E^*. For this, we view the random process of sampling the indices in I by the verifier as follows. It first chooses $t/2$ indices, call this I_F (the first half). Then, it chooses the remaining $t/2$ indices, denote by I_S (the second half).

Next we bound the required probability via the following events,

- The probability that $|I_F \cap \{i_1, \ldots, i_{t/6}\}| \geq t/1000$. Using an union bound, this is at most

$$\begin{aligned}\binom{\frac{t}{2}}{\frac{t}{2}-\frac{t}{1000}}\left(\frac{N-\frac{t}{6}}{N}\right)^{\frac{t}{2}-\frac{t}{1000}} &= \binom{\frac{t}{2}}{\frac{t}{1000}}\left(\frac{4t-\frac{t}{6}}{4t}\right)^{\frac{t}{2}-\frac{t}{1000}} \\ &< \left(\frac{1000e}{2}\right)^{\frac{t}{1000}}\left(\frac{23}{24}\right)^{\frac{t}{4}}\left(\frac{23}{24}\right)^{\frac{t}{8}} \\ &= \left(\left(\frac{1000e}{2}\right)^{1/1000}\left(\frac{23}{24}\right)^{1/4}\right)^{t}\left(\frac{23}{24}\right)^{\frac{t}{8}} \\ &< \left(\frac{23}{24}\right)^{\frac{t}{8}}\end{aligned}$$

  which is negligible. Therefore, except with negligible probability $|I_F \cap \{i_1, \ldots, i_{t/6}\}| > t/18$. Let us denote the subscripts in $\{i_1, \ldots, i_{t/6}\}$ that are part of the intersection $I_F \cap \{i_1, \ldots, i_{t/6}\}$ by Γ.
- Conditioned on $|\Gamma| > t/1000$, the probability that $|I_S \cap \{\tilde{i}_j\}_{j \in \Gamma}| \geq \frac{t}{10^6}$ can be bounded using another union bound and shown to be negligible. [15]

This means that, except with negligible probability, there are at least $t/10^6$ pairs $(i_j, \tilde{i}_j)$ such that both of them are in I. The prover is now caught if the $ch_{i_j} = b_{i_j}$ and $ch_{\tilde{i}_j} = b_{\tilde{i}_j}$ which occurs with probability $\frac{1}{4}$ for every pair. The probability that this does not happen for all $t/10^6$ pairs is at most $\left(\frac{3}{4}\right)^{-t/10^6}$ which is negligible.

The completeness and zero-knowledge follows analogously to the previous section. □

6.3. *Instantiating $\mathcal{F}_{\text{COM}}$*

We obtain two corollaries depending on how we instantiate both the commitments of the verifier and the prover. The first instantiation is based on [34] scheme where the commitment made by the verifier in the first message is computed using a statistically hiding commitment scheme, whereas the commitment made by the prover is computed using a statistically binding scheme. This yields a 5-round commit-and-prove zero-knowledge proof. More formally, we obtain the following corollary.

[15] We have not optimized the parameters as our focus is to demonstrate theoretical feasibility of such protocols.

Corollary 6.4. *Assume the existence of collision-resistant hash-functions. Then, there exists a 5-round input-delayed commit-and-prove zero-knowledge proof where the underlying primitive and* NP*-relation are used in a black-box way.*

The second instantiation is only based on one-way functions, where the verifier commits to its challenge using a perfectly binding commitment and the prover commits to its message using the 3-round parallel extractable commitment scheme of [68]. This yields a 6-round protocol based on one-way functions. More precisely, we obtain the following corollary.

Corollary 6.5. *Assume the existence of one-way functions. Then, there exists a 6-round input-delayed commit-and-prove zero-knowledge argument where the underlying primitive and* NP*-relation are used in a black-box way.*

7. Adaptive Instance-Dependent Commitments for the Binary Message Space

In the current and next 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 Sect. 3.5) and robust randomized encoding (see Sect. 3.6), where the former can be viewed as a warmup of the latter.

7.1. *Adaptive 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 Sect. 3.5 which, in turn, implies a construction for the binary message space $\{0, 1\}$ based on one-way functions (see more detailed discussion in Sect. 3.5). 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 $(x, \omega) \in \mathcal{R}$.[16] 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 encoding information, 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 to the input encoding $\tilde{z}$ that is output by $\mathsf{SimGC}(\mathrm{C}, 1)$. The decommitment to the bit 0 requires revealing the encoding information (namely, all input labels) 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

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

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 $(\widetilde{\mathrm{C}}, \mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C})$ and sends $\widetilde{\mathrm{C}}, \mathsf{dk}$ and $\sigma = \mathsf{com}(\mathsf{ek})$ to the receiver.

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

- $\mathsf{Com}(x, 1)$: S generates $(\widetilde{\mathrm{SimC}}, \tilde{z}, \mathsf{dk}) \leftarrow \mathsf{SimGC}(1^\kappa, \mathrm{C}, 1)$ and sends $\widetilde{\mathrm{SimC}}, \mathsf{dk}$ 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 $\mathsf{Dec}(\mathsf{dk}, \tilde{y})$ 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 ek, denoted by $\tilde{\omega}$.

Fig. 3. Instance-dependent commitment from garbled schemes.

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 garbling constructions from the literature, such as the [61] scheme, we note that the communication complexity of our construction for committing a single bit equals $O(s \cdot \mathsf{poly}(\kappa))$ where s is the circuit's size and κ is the security parameter.

We are now ready to formally describe our construction in Fig. 3. We prove the following theorem,

Theorem 7.1. *Assume the existence of one-way permutations. Then, the protocol presented in Fig. 3 is a secure adaptive instance-dependent commitment scheme for any language in NP.*

Proof. The proof follows by demonstrating the three properties from Definition 3.4.

Computational hiding Toward 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 detail, recall that a commitment to 0 is a garbling of C and a commitment to ek, 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 ek 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 ek 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

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

and

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

where $\mathbf{y} = 1$ in our case and $[\mathsf{SimGC}\,(1^\kappa, \mathrm{C}, \mathbf{y})]_i$ denotes the ith output of algorithm SimGC. The first indistinguishability proof is reduced to the hiding property of the commitment scheme, where a commitment to ek 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 an external string (that might be either be a commitment to ek 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 $(\widetilde{\mathrm{SimC}}, \mathsf{dk}, r, \sigma)$, whereas the second distribution corresponds to an execution by the oblivious sampler which yields $(\widetilde{\mathrm{C}}, \mathsf{dk}, r', \sigma)$. By the oblivious sampling property specified in Sect. 3.5, 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 $(\widetilde{\mathrm{C}}, \mathsf{dk}, \sigma)$ denote a commitment to 0 as specified in Fig. 3. Then, $(\widetilde{\mathrm{C}}, \mathsf{dk}, \sigma)$ 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, equivocation to 1 is not possible. Moreover, if the commitment is comprised from $(\widetilde{\mathrm{SimC}}, \mathsf{dk}, \sigma)$, 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. □

We note that our construction can be based on one-way functions if we use the two-round Naor [64] commitment scheme instead of a non-interactive commitment scheme based on one-way permutations. Therefore, we obtain the following corollary.

Corollary 7.2. *Assume the existence of one-way functions. Then, there exists a two-round adaptive instance-dependent commitment scheme for any language in NP.*

7.2. *Adaptive Instance-Dependent Commitments from Robust RE*

Our second instance-dependent construction is based on robust randomized encoding that maintains oblivious sampling, formally defined in Sect. 3.6. On a high level, our instance-dependent commitment scheme from randomized encoding will follow the same approach as in Sect. 7.1. We begin with a randomized encoding for the following function f: $f(x, \omega) = \mathcal{R}(x, \omega)$. Now, since the randomized encoding is robust, in the sense of Definition 3.11, we can split the simulation algorithm to offline and online parts, namely S_{OFF} and S_{ON}, where S_{OFF} on input x and randomness r' outputs the offline part of the encoding s_{OFF} and S_{ON} on input $(1, r')$ outputs the online part s_{ON}. Our complete construction is described in Fig. 4.

We reprove theorem with the commitment scheme described in Fig. 4. As above, the proof follows by demonstrating the three properties from Definition 3.4 and follows very similarly. Informally, the hiding property follows due to the privacy of the randomized encoding and the pseudorandomness property of com, as the differences between a commitment to 0 and 1 are by first either invoking the real encoding algorithm or the simulator, as well as either committing to a valid randomness r or sampling the commitment at random. Next, adaptivity follows from the fact that a fake commitment of 0, can be explained as a commitment to 1 by exploiting the oblivious sampling property of the randomized encoding which allows to explain a real encoding as a simulated one, as well as the ability to explain a commitment for com as obliviously generated. Finally, binding follows from the prefect robustness property of the randomized encoding, for which given a valid offline encoding it is not possible to produce an online encoding that makes the decoder output 1.

Instance-Dependent Commitment Scheme from Robust Randomized Encoding

Building block: Pseudorandom perfectly binding commitment scheme com.

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

The commitment scheme:

- $\mathsf{Com}(x, 0)$: Sample r and output $(\widehat{f}_{\mathrm{OFF}}(r), \sigma)$ where $\sigma = \mathsf{com}(r)$.

  Decommitment: The decommitment simply contains the decommitment of σ to the value r. The verifier accepts only if the (offline) encoding was computed correctly with randomness r and the decommitment information of σ is correct. Otherwise, the verifier rejects.
- $\mathsf{Com}(x, 1)$: Compute $s_{\mathrm{OFF}} \leftarrow S_{\mathrm{OFF}}(r')$ and output $(s_{\mathrm{OFF}}, \sigma)$, where $\sigma \leftarrow \{0,1\}^t$ and $t = |\mathsf{com}(r)|$.

  Decommitment: The decommitment contains $s_{\mathrm{ON}} \leftarrow S_{\mathrm{ON}}(1, r')$ and explanation of σ as an obliviously generated commitment, i.e., a random string. The verifier computes $B(s_{\mathrm{OFF}}, s_{\mathrm{ON}})$ and accepts only if it evaluates to 1. Otherwise it rejects.
- $\mathsf{Com}'(x)$: Is identical to $\mathsf{Com}(x, 0)$, i.e., output $(\widehat{f}_{\mathrm{OFF}}(r), \sigma = \mathsf{com}(r))$.
- $\mathsf{Equiv}(x, \omega, c, 0, r)$: 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 $\widehat{f}_{\mathrm{ON}}((x, \omega), r)$ and further explains σ as an obliviously generated commitment. Recall that the receiver now checks if $B(\widehat{f}_{\mathrm{OFF}}(r), \widehat{f}_{\mathrm{ON}}((x, \omega), r)) = 1$.

Fig. 4. Instance-dependent commitment from robust randomized encoding.

7.3. *Application: 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 3.9. Moreover, consider the same commitment/decommitment algorithms for both 0 and 1 as specified in Fig. 3. 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 none 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 decommitment 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 7.3. *Assume the existence of one-way functions. Then, 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 Σ-protocols [25] and for Blum's Graph Hamiltonicity [31].

8. 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 Sect. 7. For simplicity, we focus on honest-verifier zero-knowledge proofs, which can be transformed to zero-knowledge proofs using standard tools.

8.1. *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 ek using the commitment scheme from Fig. 3. Next, upon receiving a challenge bit b from the verifier, the prover continues as follows. 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 [61,69] 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 is 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 Sect. 3.5. 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. Formally, we require the garbling scheme to have a projection function Π_{Eval} and a simulator $(\mathcal{S}_{\mathsf{Eval}}, \mathcal{S}_{\mathsf{Adapt}})$, such that for $(\widetilde{\mathrm{C}}, \mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C})$ and $\widetilde{\mathbf{x}} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathbf{x})$, we have $\mathsf{Eval}(\Pi_{\mathsf{Eval}}(\widetilde{\mathrm{C}}), \widetilde{\mathbf{x}}) = \mathrm{C}(\mathbf{x})$ and

$$\{(\widetilde{\mathrm{C}}', \widetilde{\mathbf{x}}, \mathsf{state}) \leftarrow \mathcal{S}_{\mathsf{Eval}}(1^\kappa, \mathrm{C}, y) : \mathcal{S}_{\mathsf{Adapt}}(\mathsf{state}, \mathbf{x})\}$$
$$\overset{\mathrm{c}}{\approx} \{\widetilde{\mathrm{C}}, \mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C}; r_{\mathsf{Grb}}); \widetilde{\mathbf{x}} \leftarrow \mathsf{Enc}(\mathsf{ek}, \mathbf{x}) : (r_{\mathsf{Grb}}, \widetilde{\mathbf{x}})$$

We note that the garbling scheme described in [61] 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 Fig. 5.

Theorem 8.1. *Assume the existence of one-way functions. Then, the protocol presented in Fig. 5 is an adaptively secure honest verifier zero-knowledge proof for any language in NP with soundness error* 1/2.

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. $\mathcal{P} \rightarrow \mathcal{V}$: $\mathcal{P}$ generates $(\widetilde{\mathrm{C}}, \mathbf{dk}, \mathsf{ek}) \leftarrow \mathsf{Grb}(1^\kappa, \mathrm{C})$ and sends $\mathsf{Com}(\widetilde{\mathrm{C}}, \mathbf{dk})$ and $\mathsf{Com}(\mathsf{ek})$ to the verifier (where the commitments are computed using the real commitment algorithm).
2. $\mathcal{V} \rightarrow \mathcal{P}$: The verifier sends a random challenge bit $b \leftarrow \{0, 1\}$.
3. $\mathcal{P} \rightarrow \mathcal{V}$:
   - If $b = 0$ then the prover decommits to $\widetilde{\mathrm{C}}$, $\mathbf{dk}$ and ek. 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 $\widetilde{\omega} := \mathsf{Enc}(\mathsf{ek}, \omega)$ and then decommits to the encoding of $\widetilde{\omega}$ within the commitment of ek (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 $\mathsf{Dec}(\mathbf{dk}, \widetilde{y})$ equals 1.

Fig. 5. Adaptive zero-knowledge proof for any language $\mathcal{L} \in \mathsf{NP}$.

Using our adaptive instance-dependent commitment scheme from Sect. 7.1, 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 properly 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 7.1.

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 0 from the verifier, the simulator completes the execution as would the honest prover do, decommitting to the garbled circuit, the decoding information and the secret key. Upon receiving the bit 1, the simulator decommits to the simulated garbled circuit and the simulated decoding information (that is embedded within the overall decoding information), which corresponds to the decoding labels as returned by the simulator. Then, 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 ek 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. □

References

[1] S. Ames, C. Hazay, Y. Ishai, M. Venkitasubramaniam, Ligero: lightweight sublinear arguments without a trusted setup, in *CCS* (2017), pp. 2087–2104

[2] B. Applebaum, Y. Ishai, E. Kushilevitz, Cryptography in NC^0, in *FOCS* (2004), pp. 166–175

[3] B. Applebaum, Y. Ishai, E. Kushilevitz, Cryptography in nc^0. *SIAM J. Comput.* **36**(4), 845–888 (2006)

[4] B. Applebaum, Y. Ishai, E. Kushilevitz, From secrecy to soundness: efficient verification via secure computation, in *ICALP* (2010), pp. 152–163

[5] S. Agrawal, Y. Ishai, D. Khurana, A. Paskin-Cherniavsky, Statistical randomized encodings: a complexity theoretic view, in *ICALP* (2015), pp. 1–13

[6] B. Applebaum, Y. Ishai, E. Kushilevitz, B. Waters, Encoding functions with constant online rate or how to compress garbled circuits keys, in *CRYPTO* (2013), pp. 166–184

[7] B. Applebaum, Key-dependent message security: generic amplification and completeness. *J. Cryptol.* **27**(3), 429–451 (2014)

[8] G. Brassard, D. Chaum, C. Crépeau, Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.* **37**(2), 156–189 (1988)

[9] D. Beaver, Correlated pseudorandomness and the complexity of private computations, in *STOC* (1996), pp. 479–488

[10] M. Ben-Or, S. Goldwasser, A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract), in *STOC* (1988), pp. 1–10

[11] B. Barak, I. Haitner, D. Hofheinz, Y. Ishai, Bounded key-dependent message security, in *EUROCRYPT* (2010), pp. 423–444

[12] M. Bellare, V. T. Hoang, P. Rogaway, Foundations of garbled circuits, in *CCS* (2012), pp. 784–796

[13] M. Bellare, S. Micali, R. Ostrovsky, Stoc., 482–493 (1990)

[14] D. Beaver, S. Micali, P. Rogaway, The round complexity of secure protocols (extended abstract), in *STOC* (1990), pp. 503–513

[15] R. Canetti, Security and composition of multiparty cryptographic protocols. *J. Cryptol.* **13**(1), 143–202 (2000)

[16] D. Chaum, C. Crépeau, I. Damgård, Multiparty unconditionally secure protocols (abstract), in *CRYPTO* (1987), p. 462

[17] R. Canetti, I. Damgård, S. Dziembowski, Y. Ishai, T. Malkin, Adaptive versus non-adaptive security of multi-party protocols. *J. Cryptol.* **17**(3), 153–207 (2004)

[18] I. Cascudo, I. Damgård, B. M. David, I. Giacomelli, J. B. Nielsen, R. Trifiletti, Additively homomorphic UC commitments with optimal amortized overhead, in *PKC* (2015), pp. 495–515
[19] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, G. Zaverucha, Post-quantum zero-knowledge and signatures from symmetric-key primitives, in *CCS* (2017), pp. 1825–1842
[20] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, Universally composable two-party and multi-party secure computation, in *STOC* (2002), pp. 494–503
[21] M. Ciampi, G. Persiano, A. Scafuro, L. Siniscalchi, I. Visconti, Improved or-composition of sigma-protocols, in *TCC* (2016), pp. 112–141
[22] M. Ciampi, G. Persiano, A. Scafuro, L. Siniscalchi, I. Visconti, Online/offline OR composition of sigma protocols, in *EUROCRYPT* (2016), pp. 63–92
[23] R. Canetti, O. Poburinnaya, M. Venkitasubramaniam, Equivocating yao: constant-round adaptively secure multiparty computation in the plain model, in *STOC* (2017), pp. 497–509
[24] C. Crépeau, J. van de Graaf, A. Tapp, Committed oblivious transfer and private multi-party computation, in *CRYPTO* (1995), pp. 110–123
[25] I. Damgård, On Σ-protocols. http://www.cs.au.dk/~ivan/Sigma.pdf (2010)
[26] I. Damgård, Y. Ishai, Scalable secure multiparty computation, in *CRYPTO* (2006), pp. 501–520
[27] I. Damgård, J. B. Nielsen, Improved non-committing encryption schemes based on a general complexity assumption, in *CRYPTO* (2000), pp. 432–450
[28] I. Damgård, T. P. Pedersen, B. Pfitzmann, On the existence of statistically hiding bit commitment schemes and fail-stop signatures, in *CRYPTO* (1993), pp. 250–265
[29] U. Feige, J. Kilian, M. Naor, A minimal model for secure computation (extended abstract), in *STOC* (1994), pp. 554–563
[30] U. Feige, D. Lapidot, A. Shamir, Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput*. **29**(1), 1–28 (1999)
[31] U. Feige, A. Shamir, Zero knowledge proofs of knowledge in two rounds, in *CRYPTO* (1989), pp. 526–544
[32] R. Gennaro, C. Gentry, B. Parno, Non-interactive verifiable computing: Outsourcing computation to untrusted workers, in *CRYPTO* (2010), pp. 465–482
[33] V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, A. Wadia, Founding cryptography on tamper-proof hardware tokens, in *TCC* (2010), pp. 308–326
[34] O. Goldreich, A. Kahan, How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptol*. **9**(3), 167–190 (1996)
[35] C. Ganesh, Y. Kondi, A. Patra, P. Sarkar, Efficient adaptively secure zero-knowledge from garbled circuits, in *PKC* (2018), pp. 499–529
[36] S. Goldwasser, Y. T. Kalai, G. N. Rothblum, One-time programs, in *CRYPTO* (2008), pp. 39–56
[37] V. Goyal, C.-K. Lee, R. Ostrovsky, I. Visconti, Constructing non-malleable commitments: a black-box approach, in *FOCS* (2012), pp. 51–60
[38] I. Giacomelli, J. Madsen, C. Orlandi, Zkboo: faster zero-knowledge for boolean circuits, in *USENIX* (2016), pp. 1069–1083
[39] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems. *SIAM J. Comput*. **18**(1), 186–208 (1989)
[40] O. Goldreich, S. Micali, A. Wigderson, How to play any mental game or A completeness theorem for protocols with honest majority, in *STOC* (1987), pp. 218–229
[41] O. Goldreich, *Foundations of Cryptography: Basic Tools* (Cambridge University Press, Cambridge, 2001)
[42] V. Goyal, R. Ostrovsky, A. Scafuro, I. Visconti, Black-box non-black-box zero knowledge, in *STOC* (2014), pp. 515–524
[43] J. A. Garay, D. Wichs, H.-S. Zhou, Somewhat non-committing encryption and efficient adaptively secure oblivious transfer, in *CRYPTO* (2009), pp. 505–523
[44] D. Harnik, Y. Ishai, E. Kushilevitz, J. B. Nielsen, Ot-combiners via secure computation, in *TCC* (2008), pp. 393–411
[45] B. Hemenway, Z. Jafargholi, R. Ostrovsky, A. Scafuro, D. Wichs, Adaptively secure garbled circuits from one-way functions, in *CRYPTO* (2016), pp. 149–178

[46] S. Halevi, S. Micali, Practical and provably-secure commitment schemes from collision-free hashing, in *CRYPTO* (1996), pp. 201–215
[47] I. Haitner, O. Reingold, A new interactive hashing theorem, in *CCC* (2007), pp. 319–332
[48] Y. Ishai, E. Kushilevitz, Randomizing polynomials: a new representation with applications to round-efficient secure computation, in *FOCS* (2000), pp. 294–304
[49] Y. Ishai, E. Kushilevitz, Perfect constant-round secure computation via perfect randomizing polynomials, in *ICALP* (2002), pp. 244–256
[50] Y. Ishai, E. Kushilevitz, R. Ostrovsky, A. Sahai, Zero-knowledge from secure multiparty computation, in *STOC* (2007), pp. 21–30
[51] Y. Ishai, E. Kushilevitz, R. Ostrovsky, A. Sahai, Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput*. **39**(3), 1121–1152 (2009)
[52] Y. Ishai, E. Kushilevitz, M. Prabhakaran, A. Sahai, C.-H. Yu, Secure protocol transformations, in *CRYPTO* (2016), pp. 430–458
[53] T. Itoh, Y. Ohta, H. Shizuya, A language-dependent cryptographic primitive. *J. Cryptol*. **10**(1), 37–50 (1997)
[54] Y. Ishai, M. Prabhakaran, A. Sahai, Founding cryptography on oblivious transfer—efficiently, in *CRYPTO* (2008), pp. 572–591
[55] Y. Ishai, M. Prabhakaran, A. Sahai, Secure arithmetic computation with no honest majority, in *TCC* (2009), pp. 294–314
[56] Y. Ishai, M. Weiss, Probabilistically checkable proofs of proximity with zero-knowledge, in *TCC* (2014), pp. 121–145
[57] Z. Jafargholi, A. Scafuro, D. Wichs, Adaptively indistinguishable garbled circuits, in *TCC* (2017), pp. 40–71
[58] Z. Jafargholi, D. Wichs, Adaptive security of yao's garbled circuits, in *TCC* (2016), pp. 433–458
[59] J. Kilian, Founding cryptography on oblivious transfer, in *STOC* (1988), pp. 20–31
[60] J. Katz, R. Ostrovsky, Round-optimal secure two-party computation, in *CRYPTO* (2004), pp. 335–354
[61] Y. Lindell, B. Pinkas, A proof of security of Yao's protocol for two-party computation. *J. Cryptol*. **22**(2), 161–188 (2009)
[62] D. Lapidot, A. Shamir, Publicly verifiable non-interactive zero-knowledge proofs, in *CRYPTO* (1990), pp. 353–365
[63] Y. Lindell, H. Zarosim, Adaptive zero-knowledge proofs and adaptively secure oblivious transfer. *J. Cryptol*. **24**(4), 761–799 (2011)
[64] M. Naor, Bit commitment using pseudorandomness. *J. Cryptol*. **4**(2), 151–158 (1991)
[65] R. Ostrovsky, A. Scafuro, M. Venkitasubramaniam, Resettably sound zero-knowledge arguments from OWFs: the (semi) black-box way, in *TCC* (2015), pp. 345–374
[66] S. J. Ong, S. P. Vadhan, An equivalence between zero knowledge and commitments, in *TCC* (2008), pp. 482–500
[67] B. Pinkas, T. Schneider, N. P. Smart, S. C. Williams, Secure two-party computation is practical, in *ASIACRYPT* (2009), pp. 250–267
[68] R. Pass, H. Wee, Black-box constructions of two-party protocols from one-way functions, in *TCC* (2009), pp. 403–418
[69] A. C.-C. Yao, How to generate and exchange secrets (extended abstract), in *FOCS* (1986), pp. 162–167
[70] Y. Lindell, H. Zarosim, Adaptive zero-knowledge proofs and adaptively secure oblivious transfer, in *TCC* (2009), pp. 183–201