J. Cryptol. (2011) 24: 761–799
DOI: 10.1007/s00145-010-9072-z

Journal of
CRYPTOLOGY

Adaptive Zero-Knowledge Proofs and Adaptively Secure Oblivious Transfer

Yehuda Lindell and Hila Zarosim

Dept. of Computer Science, Bar-Ilan University, Ramat-Gan, Israel
lindell@cs.biu.ac.il; zarosih@cs.biu.ac.il

Communicated by Ran Canetti

Received 22 July 2009
Online publication 28 July 2010

Abstract. In the setting of secure computation, a set of parties wish to securely compute some function of their inputs, in the presence of an adversary. The adversary in question may be *static* (meaning that it controls a predetermined subset of the parties) or *adaptive* (meaning that it can choose to corrupt parties during the protocol execution and based on what it sees). In this paper, we study two fundamental questions relating to the basic zero-knowledge and oblivious transfer protocol problems:

- *Adaptive zero-knowledge proofs:* We ask whether it is possible to construct adaptive zero-knowledge *proofs* (with unconditional soundness) for all of $\mathcal{NP}$. Beaver (STOC 1996) showed that known zero-knowledge proofs are not adaptively secure, and in addition showed how to construct zero-knowledge arguments (with computational soundness).
- *Adaptively secure oblivious transfer:* All known protocols for adaptively secure oblivious transfer rely on seemingly stronger hardness assumptions than for the case of static adversaries. We ask whether this is inherent, and in particular, whether it is possible to construct adaptively secure oblivious transfer from enhanced trapdoor permutations alone.

We provide surprising answers to the above questions, showing that achieving adaptive security is sometimes harder than achieving static security, and sometimes not. First, we show that assuming the existence of one-way functions only, there exist adaptive zero-knowledge proofs for all languages in $\mathcal{NP}$. In order to prove this, we overcome the problem that all adaptive zero-knowledge protocols known until now used equivocal commitments (which would enable an all-powerful prover to cheat). Second, we prove a black-box separation between adaptively secure oblivious transfer and enhanced trapdoor permutations. As a corollary, we derive a black-box separation between adaptively and statically secure oblivious transfer. This is the first black-box separation to relate to adaptive security and thus the first evidence that it is indeed harder to achieve security in the presence of adaptive adversaries than in the presence of static adversaries.

Key words. Adaptive security, Zero-knowledge proofs, Oblivious transfer, Black-box separations, Minimal assumptions.

This research was supported by the ISRAEL SCIENCE FOUNDATION (grant No. 781/07).

1. Introduction

In the setting of secure two-party and multiparty computation, parties with private inputs wish to securely compute some joint function of their inputs, where "security" must hold in the presence of adversarial behavior by some of the parties. An important parameter in any definition of security relates to the adversary's power. Is the adversary computationally bounded or all powerful? Is the adversary semi-honest (meaning that it follows all protocol instructions but tries to learn more than it's supposed to by analyzing the messages it receives) or is it malicious (meaning that it can arbitrarily deviate from the protocol specification)? Finally, are the adversarial corruptions *static* (meaning that the set of corrupted parties is fixed) or *adaptive* (meaning that the adversary can corrupt parties throughout the computation and the question of who to corrupt and when may depend on the adversary's view in the protocol execution). It is desirable to achieve security in the presence of adaptive adversaries where possible, since it models the real-world phenomenon of "hackers" actively breaking into computers, possibly while they are executing secure protocols. However, it seems to be technically harder to achieve security in the presence of adaptive adversaries. Among other things, it requires the ability to construct a simulator who can first generate a transcript blindly (without knowing any party's input) and then later, upon receiving inputs, "explain" the transcript as an execution of honest parties with those inputs.

In this paper, we ask two basic questions related to the feasibility of achieving security in the presence of adaptive adversaries. Our questions were borne out of the following two observations:

1. *Adaptive zero-knowledge proofs:* It has been shown that the zero-knowledge proof system of [24] (and all others known) for $\mathcal{NP}$-complete languages is not secure in the presence of adaptive adversaries, or else the polynomial hierarchy collapses [1]. Due to this result, all known zero-knowledge protocols for $\mathcal{NP}$-complete languages in the adaptive setting are *arguments*, meaning that soundness only holds in the presence of a polynomial-time prover (adaptive zero-knowledge arguments were presented by [1] and later in the context of universal composability; e.g., see [7,9]). However, the question of whether or not adaptive zero-knowledge *proofs* exist for all of $\mathcal{NP}$ has not been addressed.
2. *Adaptively secure oblivious transfer:* One of the goals of the theory of cryptography is to understand what assumptions are necessary and sufficient for carrying out cryptographic tasks; see, for example, [26]. Despite this, no such study has been carried out regarding adaptively secure protocols. In particular, we do not know what assumptions are necessary for achieving adaptively secure oblivious transfer (since oblivious transfer is complete for secure computation, this question has important ramifications to adaptively secure computation in general). Currently, what is known is that although statically secure oblivious transfer can be constructed from enhanced trapdoor permutations [15,23], all constructions for adaptively secure oblivious transfer use additional assumptions like the ability to sample a permutation without knowing its trapdoor [3,9].

Our Results—Adaptive Zero-Knowledge Proofs All known zero-knowledge protocols for $\mathcal{NP}$ essentially follow the same paradigm: the prover sends the verifier commitments that are based on the statement being proved (and its witness), and the verifier

then asks the prover to open part or all of the commitments. Based on the prover's answer, the verifier is either convinced that the statement is true or detects the prover cheating. It therefore follows that soundness only holds if the commitment scheme used is *binding*, and this is a problem in the setting of adaptive security. Consider an adversary that corrupts the verifier at the beginning of the execution and the prover at the end. In this case, the zero-knowledge simulator must generate a transcript without knowing the NP-witness. However, at the end, after the prover is corrupted (and the simulator then receives a witness), it must be able to show that the commitments were generated using that witness. Until now, this has been solved by using *equivocal* commitments that can be opened to any value desired (in order for soundness to hold, the ability to equivocate is given to the simulator and not the real prover). However, equivocability needed by the simulator for the 'yes' instances allows an all-powerful prover to break binding and thus soundness on 'no' instances and this means that the protocol has only computational soundness. Indeed, the above observation led us to initially conjecture that adaptive zero-knowledge proofs exist only for $\mathcal{SZK}$. However, our conjecture was wrong, and in this paper we prove the following theorem:

Theorem 1. *Assuming the existence of one-way functions that are hard to invert for non-uniform adversaries, there exist adaptive zero-knowledge proofs for all of* $\mathcal{NP}$.

We prove Theorem 1 by constructing a new type of *instance-dependent* commitment scheme. Instance-dependent commitment schemes are commitments whose properties depend on whether the instance (or statement) in question is in the language or not [4,28]. Typically, they are defined for a language L as follows. Let x be a statement. If $x \in L$ then the commitment associated with x is computationally hiding and if $x \notin L$ then the commitment associated with x is perfectly binding. This has proven very useful in the context of zero-knowledge where hiding alone is needed for the case of $x \in L$, and binding alone is needed in the case of $x \notin L$; see, for example, [33,34,41]. We construct an instance-dependent commitment scheme with the additional property that if $x \in L$ then the commitment is *equivocal* and the simulator can open it to any value it wishes. To be more exact, we need the commitment itself to be *adaptively secure*, meaning that it must be possible to generate a commitment value c and then later find "random coins" r for any bit b so that c is a commitment string generated by an honest committer with input b and random coins r.[1] In contrast to the above, if $x \notin L$ then the commitment is still perfectly binding. Given such a commitment (which is actually very similar to the commitment schemes presented in [16] and [9]) we are able to construct the first computational zero-knowledge *proof* for all of $\mathcal{NP}$ that is secure also in the case of adaptive corruptions.[2]

[1] We stress that this is a strictly stronger requirement than equivocality. In most equivocal commitments, the committer reveals only some of its coins upon decommitting. This does not suffice for achieving adaptive commitments.

[2] In [34], adaptively secure commitment schemes were constructed for the languages of Graph Isomorphism and Quadratic Residuosity (although they were not presented in this way nor for this purpose). The constructions in [34] are incomparable to ours. On the one hand, they require no hardness assumptions whereas we use one-way functions. On the other hand, our construction is for all languages in $\mathcal{NP}$ whereas they are restricted to the above two specific languages (which are also in $\mathcal{SZK}$).

Our Results—Adaptively Secure Oblivious Transfer As we have mentioned, all known protocols for adaptively secure oblivious transfer require assumptions of the flavor that it is possible to sample a permutation without its trapdoor. In contrast, standard trapdoor permutations do not have this property. We remark that enhanced trapdoor permutations do have the property that it is possible to sample an element in the domain of the permutation without knowing its preimage. This begs the question as to whether such "oblivious sampling" of the permutation's domain suffices for achieving adaptively secure oblivious transfer, or is something stronger needed (like oblivious sampling of permutations themselves). We remark that oblivious sampling is used in this context by having the simulator sample unobliviously and then "lie" in its final transcript by claiming to have sampled in the regular way. However, this strategy is problematic when the oblivious sampling is carried out on elements in the domain because if the trapdoor is known then it may be possible to see if the preimage of the sampled value appears implicitly in the protocol transcript. (For example, in the protocol of [15], the preimages fully define the sender's input and so if the trapdoor is known, the values can be checked.) Of course, such arguments do not constitute any form of evidence. In order to demonstrate hardness, we use the methodology of black-box separations, introduced by [27] and later used in [19,31,39,40] amongst others. We prove the following informally stated theorem:

Theorem 2. *There exists an oracle relative to which enhanced trapdoor permutations exist but adaptively secure oblivious transfer does not exist.*

Recalling that statically secure oblivious transfer can be constructed from any enhanced trapdoor permutation in a black-box way [15,23], we obtain the following corollary:

Corollary 3. *There exists an oracle relative to which statically secure oblivious transfer exists but adaptively secure oblivious transfer does not exist.*

This is the first evidence that it is strictly harder to achieve security in the presence of adaptive adversaries than to achieve security in the presence of static adversaries. We prove Theorem 2 by showing that if it is possible to achieve adaptively secure oblivious transfer using only enhanced trapdoor permutations, then it is possible to achieve statically secure oblivious transfer using only symmetric encryption (this is very inexact but sufficient for intuition). We then show that statically secure oblivious transfer does not exist relative to most symmetric encryption oracles. In order to prove this, we use the recent result of [11] that shows the equivalence of the random oracle and ideal cipher models, to replace a symmetric encryption oracle by a "plain" random oracle (using six rounds of the Luby–Rackoff construction [32]).[3] This enables us to extend the black-box separation of [27] to show that key agreement does not exist relative to most symmetric encryption oracles. (Indeed this is a novel interpretation of that result and it means that all black-box separations with random oracles hold also with symmetric

[3] The reason that 6 rounds are needed and not 4 (as in the construction of pseudorandom permutations from pseudorandom functions) is due to the fact that the distinguisher has access to the intermediate values.

encryption). We conclude the proof by recalling that key agreement can be constructed from oblivious transfer [19]. Thus, adaptively secure oblivious transfer cannot be constructed in a black-box way from enhanced trapdoor permutations. We remark that all of our results for oblivious transfer are proven for semi-honest adversaries (and thus hold also for malicious adversaries).

Our proof makes no explicit use of the fact that the functionality being computed is oblivious transfer and holds for any functionality. We conclude that either a given function can be securely computed statically assuming only the existence of one-way functions (or to be more exact, only given a "symmetric" random oracle), or enhanced trapdoor permutations do not suffice for computing it with adaptive security.

Related Work Instance-dependent commitment schemes were first implicitly used in [4] to construct a constant round zero-knowledge proof for the language of Graph Isomorphism. In [28], they follow these ideas and explicitly define instance-dependent commitment schemes[4] as commitment schemes where both the sender and the receiver receive an instance $x \in \{0,1\}^*$ and the binding and the hiding properties depend on whether $x \in L$ or not. Since then, a great deal of work has been carried out to investigate the relationship between zero-knowledge proofs and instance-dependent commitment schemes (see [12,13,29,33,36,41]) and finally in the recent work of [37] it was shown that instance-dependent commitment schemes are necessary and sufficient for constructing zero-knowledge proofs. It is interesting to note that in contrast to regular commitment schemes, where the commitment cannot be both statistically hiding and statistically binding, instance-dependent commitment schemes that are statistically hiding when $x \in L$ and statistically binding when $x \notin L$ exist for certain languages (such as Graph Isomorphism).

The method of proving black-box separations between cryptographic primitives, as a way to conclude that it is not likely that the existence of a certain cryptographic primitive implies the existence of another primitive, was first introduced in the seminal work of Impagliazzo and Rudich [27]. In their work, they consider a world where all parties have access to a random permutation oracle, which is provably one-way in the strongest sense. On the other hand, they show that if $\mathcal{P} = \mathcal{NP}$, there doesn't exist a secure key agreement protocol (even relative to such a random oracle). This result has the following consequence: There is an oracle relative to which one-way permutation exists but key agreement does not exist (This oracle is constructed from the random oracle and a $\mathcal{PSPACE}$-complete problem). This method of proving black-box separation was subsequently used in many other works (see [10,17,18,20,21,30,31,38,40]).

Adaptive zero-knowledge arguments were constructed in [1] and later adaptively secure universally composable zero-knowledge arguments were constructed in [7,9]. Adaptively secure oblivious transfer was constructed in [3] using trapdoor permutations with the additional property that it is possible to select a permutation without knowing its trapdoor, and in [9] from non-committing encryptions (see [2,8,14]). All constructions of adaptively secure oblivious transfer (and non-committing encryptions) require the possibility to sample a permutation without its trapdoor. In this sense, our result is a

[4] Actually, in [28], they use the term "language-dependent cryptographic primitives".

complementary result to these construction as we prove that enhanced trapdoor permutations alone are not sufficient for constructing an adaptively secure oblivious transfer in a black-box manner.

In [11], it is shown that the random oracle and the ideal cipher models are equivalent. Our techniques imply a new application for this result: all black-box separations that have been proven relative to a random oracle also hold relative to an ideal cipher (or a symmetric encryption oracle).

2. Preliminaries and Definitions

2.1. *Preliminaries and Notations*

We let n denote the security parameter. We say that a function $\mu : \mathbb{N} \to \mathbb{N}$ is negligible if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\mu(n) < \frac{1}{p(n)}$. We use the abbreviation PPT to denote probabilistic polynomial-time. For an $\mathcal{NP}$ relation R, we denote by R_x the set of witnesses of x and by L_R its associated language. That is, $R_x = \{w | \ (x, w) \in R\}$ and $L_R = \{x | \ \exists w \ \ (x, w) \in R\}$.

Let $\langle A, B\rangle$ be an interactive protocol. $\langle A(x_A; r_A), B(x_B; r_B)\rangle$ denotes the joint output of A and B, where the input of A is x_A and its random tape is r_A and the input of B is x_B and its random tape is r_B. $\langle A(x_A), B(x_B)\rangle$ denotes the random variable describing $\langle A(x_A; r_A), B(x_B; r_B)\rangle$ where the random tapes of the parties are chosen uniformly. The view here contains the oracle replies as well.

$\text{VIEW}_A^{\Pi}(x_A, x_B)$ is a random variable describing the view of A in an execution of protocol Π on inputs x_A and x_B, where the random tapes are chosen uniformly. The view of a party includes its input, random tape and the messages the party received. For an oracle O, $\text{VIEW}_A^{\Pi,O}(x_A, x_B)$ describes the view of A in an execution of protocol Π on inputs x_A and x_B with oracle access to O, where the random tapes are chosen uniformly. For a distribution $\mathcal{D}$ on oracles, $\text{VIEW}_A^{\Pi,\mathcal{D}}(x_A, x_B)$ describes the view of A in an execution of protocol Π on inputs x_A and x_B where the random tapes are chosen uniformly and the parties have access to an oracle that was chosen according to $\mathcal{D}$.

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

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

Definition 2.2. Let $X = \{X(a, n)\}_{a\in\{0,1\}^*, n\in\mathbb{N}}$ and $Y = \{Y(a, n)\}_{a\in\{0,1\}^*, n\in\mathbb{N}}$ be two distribution ensembles. We say that X and Y are *computationally indistinguishable relative to an oracle* O, denoted $X \overset{c^O}{\equiv} Y$, if for every PPT oracle machine D, every $a \in \{0, 1\}^*$, every positive polynomial $p(\cdot)$ and all sufficiently large n:

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

We sometimes use the same notation when we refer to a distribution $\mathcal{D}$ over oracles rather than a single oracle O. We now present definitions for semi-honest security for static adversaries. This will be needed in our proof in Sect. 4.

Definition 2.3. Let $\langle S, R\rangle$ be an interactive protocol, where the input of S is a pair of strings $s_0, s_1 \in \{0,1\}^k$ (where $k = \text{poly}(n)$) and the input of R is a bit σ, and let n be the security parameter. We say that $\langle S, R\rangle$ *computes the* OT_1^2 *functionality*[5] if there exists a negligible function $\text{neg}(\cdot)$ such that for all n, for every $s_0, s_1 \in \{0,1\}^k$, $\sigma \in \{0,1\}$,

$$\Pr\left[\left\langle S\left(1^n, s_0, s_1\right), R\left(1^n, \sigma\right)\right\rangle = (\lambda, s_\sigma)\right] \geq 1 - \text{neg}(n).$$

We say that $\langle S^O, R^O\rangle$ computes the OT_1^2 functionality relative to an oracle O if there exists a negligible function $\text{neg}(\cdot)$ such that for every n and every s_0, s_1, σ,

$$\Pr\left[\left\langle S^O\left(1^n, s_0, s_1\right), R^O\left(1^n, \sigma\right)\right\rangle = (\lambda, s_\sigma)\right] \geq 1 - \text{neg}(n).$$

Definition 2.4. Let $\Pi = \langle S, R\rangle$ be a protocol for computing the OT_1^2 functionality. We say that Π *securely computes the* OT_1^2 *functionality in the presence of static semi-honest adversaries* if there exist two probabilistic polynomial-time algorithms $\mathcal{S}_S$ and $\mathcal{S}_R$ such that

$$\left\{\mathcal{S}_S\left(1^n, s_0, s_1\right)\right\}_{s_0,s_1\in\{0,1\}^k,\sigma\in\{0,1\}} \overset{\text{c}}{\equiv} \left\{\text{VIEW}_S^{\Pi}\left(1^n, s_0, s_1, \sigma\right)\right\}_{s_0,s_1\in\{0,1\}^k,\sigma\in\{0,1\}},$$

$$\left\{\mathcal{S}_R\left(1^n, \sigma, s_\sigma\right)\right\}_{s_0,s_1\in\{0,1\}^k,\sigma\in\{0,1\}} \overset{\text{c}}{\equiv} \left\{\text{VIEW}_R^{\Pi}\left(1^n, s_0, s_1, \sigma\right)\right\}_{s_0,s_1\in\{0,1\}^k,\sigma\in\{0,1\}}.$$

Let $\Pi = \langle S^O, R^O\rangle$ be a protocol for computing the OT_1^2 functionality relative to an oracle O. We say that Π *securely computes the* OT_1^2 *functionality relative to an oracle* O *in the presence of static semi-honest adversaries* if there exist two PPT oracle machines $\mathcal{S}_S$ and $\mathcal{S}_R$ such that

$$\left\{\mathcal{S}_S^O\left(1^n, s_0, s_1\right)\right\}_{s_0,s_1\in\{0,1\}^k,\sigma\in\{0,1\}} \overset{\text{c}^O}{\equiv} \left\{\text{VIEW}_S^{\Pi,O}\left(1^n, s_0, s_1, \sigma\right)\right\}_{s_0,s_1\in\{0,1\}^k,\sigma\in\{0,1\}},$$

$$\left\{\mathcal{S}_R^O\left(1^n, \sigma, s_\sigma\right)\right\}_{s_0,s_1\in\{0,1\}^k,\sigma\in\{0,1\}} \overset{\text{c}^O}{\equiv} \left\{\text{VIEW}_R^{\Pi,O}\left(1^n, s_0, s_1, \sigma\right)\right\}_{s_0,s_1\in\{0,1\}^k,\sigma\in\{0,1\}}.$$

2.2. *Adaptive Security—Definitions*

In this section, we provide a definition for adaptive security of two party protocols (for a deterministic functionality f). The definition is taken from [6]. Adjustments of the definition for the special cases of adaptive zero-knowledge proofs (for malicious adversaries) and adaptively secure oblivious transfer (for semi-honest adversaries) will be described afterwards.

[5] For $s_0, s_1 \in \{0,1\}^k$ and $\sigma \in \{0,1\}$, the functionality *1-out-of-2 Oblivious Transfer*, denoted OT_1^2, is defined as $\text{OT}_1^2((s_0, s_1), \sigma) = (\lambda, s_\sigma)$.

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

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_A, the information received from the environment and the corrupted parties' views of the execution.)

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

Let $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(n, x_1, x_2, z, \overrightarrow{r})$, where $\overrightarrow{r} = (r_Z, r_A, r_1, r_2)$, denote $\mathcal{Z}$'s output on input z, random tape r_Z and security parameter n after interacting with adversary $\mathcal{A}$ and parties P_1 and P_2 running protocol Π on inputs x_1, x_2, random input $\overrightarrow{r}$ and security parameter n. Let $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(n, x_1, x_2, z)$ denote the random variable describing $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(n, x_1, x_2, z, \overrightarrow{r})$ when the random tapes of the parties r_Z, r_A, r_1, r_2 are chosen uniformly. Let $\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble

$$\left\{\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(n, x_1, x_2, z)\right\}_{x_1,x_2,z\in\{0,1\}^*,n\in\mathbb{N}}.$$

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

First corruption stage: First, $\mathcal{SIM}$ receives some auxiliary information from $\mathcal{Z}$.

Next, $\mathcal{SIM}$ proceeds in at most two iterations, where in each iteration $\mathcal{SIM}$ may

decide to corrupt one of the parties. Once a party is corrupted, its input becomes known to $\mathcal{SIM}$. In addition, $\mathcal{Z}$ learns the identity of the corrupted party and hands some auxiliary information to $\mathcal{SIM}$.

Computation stage: Uncorrupted parties hand the inputs to $\mathcal{T}$. In the malicious settings, corrupted parties hand values chosen by $\mathcal{SIM}$ to $\mathcal{T}$. In the semi-honest setting, corrupted parties hand their inputs to $\mathcal{T}$. Let y_1, y_2 be the values handed to $\mathcal{T}$. $\mathcal{T}$ computes $f(y_1, y_2)$ and hands P_1 the value $f(y_1, y_1)_1$ and P_2 the value $f(y_1, y_2)_2$.

Second corruption stage: $\mathcal{SIM}$ continues in another sequence of at most two iterations, where in each iteration, $\mathcal{SIM}$ might choose to corrupt one of the parties based on its random tape and the information gathered so far. Once a party is corrupted, $\mathcal{SIM}$ learns its input, $\mathcal{Z}$ learns the identity of the corrupted party and hands $\mathcal{SIM}$ some auxiliary information.

Output: Each uncorrupted party P_i outputs $f(y_1, y_2)_i$. Corrupted parties output a special symbol $\perp$. The adversary $\mathcal{SIM}$ outputs an arbitrary function of its internal state. $\mathcal{Z}$ learns all outputs.

Postexecution corruption: After the outputs are generated, $\mathcal{SIM}$ 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 $\mathcal{SIM}$. For any such request, $\mathcal{SIM}$ generates some arbitrary answer and it might choose to corrupt any of the parties. The interaction continues until $\mathcal{Z}$ halts with an output.

Let $\text{IDEAL}_{f,\mathcal{SIM},\mathcal{Z}}(n, x_1, x_2, z, \overrightarrow{r})$, where $\overrightarrow{r} = (r_{\mathcal{Z}}, r_{\mathcal{SIM}})$, denote the output of $\mathcal{Z}$ on input z, random input $r_{\mathcal{Z}}$ and security parameter n after interacting with an ideal-process adversary $\mathcal{SIM}$ with random input $r_{\mathcal{SIM}}$, with parties having inputs (x_1, x_2) and with a trusted party $\mathcal{T}$ for evaluating the functionality f. Let $\text{IDEAL}_{f,\mathcal{SIM},\mathcal{Z}}(n, x_1, x_2, z)$ denote the random variable describing $\text{IDEAL}_{f,\mathcal{SIM},\mathcal{Z}}(n, x_1, x_2, z, \overrightarrow{r})$ where $r_{\mathcal{Z}}, r_{\mathcal{SIM}}$ are chosen uniformly. Let $\text{IDEAL}_{f,\mathcal{SIM},\mathcal{Z}}$ denote the distribution ensemble

$$\left\{\text{IDEAL}_{f,\mathcal{SIM},\mathcal{Z}}(n, x_1, x_2, z)\right\}_{x_1,x_2,z\in\{0,1\}^*,n\in\mathbb{N}}.$$

Definition 2.5. Let Π be a protocol for computing a functionality f. We say that Π *securely computes the functionality f in the presence of adaptive adversaries* if for every probabilistic polynomial-time adaptive real-life adversary $\mathcal{A}$ and every environment $\mathcal{Z}$, there exists a probabilistic polynomial-time adaptive ideal-process adversary $\mathcal{SIM}$ such that

$$\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \stackrel{c}{\equiv} \text{IDEAL}_{f,\mathcal{SIM},\mathcal{Z}}.$$

If the adversary $\mathcal{A}$ and the simulator $\mathcal{SIM}$ are restricted to semi-honest behavior, then we say that Π *securely computes the functionality f in the presence of semi-honest adaptive adversaries*.

A Special Case—Adaptive Zero-Knowledge 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 $f_R((x, w), \lambda) = (\lambda, (x, b))$ where $b = 1$ if $R(x, w) = 1$ and $b = 0$ if $R(x, w) = 0$. However, since the goal of our work is

to deal with the fundamental question of the feasibility or infeasibility of adaptive zero-knowledge proofs (as asked by Beaver [1]), we present an alternative definition that is more in line with the standard setting of zero-knowledge proof systems (that are not necessarily proofs of knowledge). The advantage of this approach is that it simplifies the proof and allows us to focus on the main issue of constructing an adaptive proof (rather than an argument), without dealing with knowledge extraction.

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

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

In the ideal process, the ideal process adversary $\mathcal{SIM}$ receives the instance x that is guaranteed to be in the language as input and interacts with the environment and corrupted parties. Thus, we only need 3 stages: first corruption stage, output stage and postexecution corruption stage (since there is no computation stage, there is also no need for a second corruption stage). We also allow the simulator to run in expected polynomial time rather than strict polynomial time (we do not know how to construct a strict polynomial-time simulator for our protocol even though it has a constant number of rounds).

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

$$\left\{\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(x, w, z)\right\}_{x\in L, w\in R_x, z\in\{0,1\}^*},$$

and $\text{IDEAL}^{ZK}_{L,\mathcal{SIM},\mathcal{Z}}$ denotes the distribution ensemble

$$\left\{\text{IDEAL}^{ZK}_{L,\mathcal{SIM},\mathcal{Z}}(x, w, z)\right\}_{x\in L, w\in R_x, z\in\{0,1\}^*}.$$

Definition 2.6. Let L be a language. We say that $\langle P, V\rangle$ is an *adaptive zero-knowledge proof system* (AZK) for L if $\langle P, V\rangle$ is an interactive proof system for L and for any PPT real-life adversary $\mathcal{A}$ and any PPT environment $\mathcal{Z}$, there exists an expected polynomial-time probabilistic adaptive ideal-process adversary $\mathcal{SIM}$ such that

$$\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \stackrel{c}{\equiv} \text{IDEAL}^{ZK}_{L,\mathcal{SIM},\mathcal{Z}}.$$

Adaptive Security Relative to Oracles In this work, we consider security relative to oracles. We therefore consider two distributions describing the real-life and the ideal-life process relative to an oracle. In this case, all parties as well as the adversary $\mathcal{A}$ and the environment $\mathcal{Z}$ are PPT oracle machines with access to the oracle O. Note that in the ideal-process the environment $\mathcal{Z}$ has access to the oracle O as well (otherwise, it is easy to distinguish the real-life model from the ideal-process) and therefore $\mathcal{SIM}$ must as well have access to the oracle O. While considering security relative to oracles, an oracle O is added to all notations (random variables, distributions, etc.).

We present the definition of security only for the semi-honest case as this suffices for our separation.

Definition 2.7. Let Π be a protocol for computing a functionality f. We say that Π *securely computes the functionality* f *in the presence of adaptive semi-honest adversaries relative to an oracle* O if for every PPT real-life adversary $\mathcal{A}$ and every PPT environment $\mathcal{Z}$, there exists a PPT adaptive ideal-process adversary $\mathcal{SIM}$, where all are given oracle access to O such that

$$\text{REAL}^{O}_{\Pi,\mathcal{A},\mathcal{Z}} \overset{c^{O}}{\equiv} \text{IDEAL}^{O}_{f,\mathcal{SIM},\mathcal{Z}}.$$

3. Adaptive Zero-Knowledge Proofs

In this section, we prove the following theorem:

Theorem 3.1. *Assuming the existence of one-way functions that are hard to invert for non-uniform adversaries, there exist adaptive zero-knowledge proofs for all of* $\mathcal{NP}$.

Specifically, we show how to construct an adaptive zero-knowledge proof for the language of Hamiltonicity (HC). Our construction is based on Blum's zero-knowledge proof for Hamiltonicity [5]. In this protocol, the prover first commits to a random permutation of the input graph G, and the verifier then chooses randomly whether to verify that the committed graph is indeed a permutation of G or that the committed graph contains a Hamiltonian cycle. Soundness holds because a non-Hamiltonian graph cannot simultaneously be a permutation of G and contain a Hamiltonian cycle. The simulator for this proof system does not know the witness and so cannot decommit to a Hamiltonian cycle after committing to a permutation of G. Therefore, it works by randomly choosing whether to send commitments to a permutation of G or to a graph containing only a random cycle of length n.

Observe that in the latter case, the commitments generated by the simulator are to different values than those generated by the real prover. This is not a problem when considering static corruptions because the hiding property of the commitments means that this cannot be distinguished. However, in the setting of adaptive corruptions, the prover can be corrupted after the simulation ends. In this case, the simulator must be able to provide random coins that demonstrate that the commitments sent initially are those that an honest prover would have sent. However, when the simulator commits to a graph containing only a Hamiltonian cycle, it cannot do this (because an honest prover never sends such a commitment). Thus, the commitment scheme used must be such that the simulator can explain commitments to 0 as commitments to 1 and vice versa (actually, it suffices that commitments to 0 be explainable as commitments to 1). One way of solving this problem is to use *equivocal* or *trapdoor* commitments. Loosely speaking, these are commitments that can be decommitted to both 0 and 1 given an appropriate trapdoor. As we have discussed, however, if we use this type of commitment scheme, then we can no longer achieve statistical soundness (since an all-powerful cheating prover can find the trapdoor and use the equivocality of the commitment scheme to fool the verifier).

We bypass this problem by constructing a new type of *instance-dependent* commitment scheme [4,28]. Roughly speaking, these are commitments whose properties depend on whether the instance in question is in the language or not. Typically, they are defined for a language L as follows. Let x be a statement. If $x \in L$ then the commitment associated with x is computationally hiding, and if $x \notin L$ then the commitment associated with x is perfectly binding. In order to achieve adaptive security, we extend this to an *adaptive* instance-dependent commitment scheme, where for $x \in L$ we have the additional property that commitments are equivocal (but for $x \notin L$ the commitments are still perfectly binding). Note that this type of commitment scheme is enough for constructing zero-knowledge proofs because the hiding and the adaptivity are essential only for proving zero-knowledge (which is needed only for $x \in L$) and the binding property is only essential for proving soundness (in the case of $x \notin L$). In Sect. 3.1, we provide a formal definition of *adaptive instance-dependent commitment schemes* along with a construction and a proof of security. In Sect. 3.2, we prove Theorem 3.1 by constructing an adaptive zero-knowledge proof system for Hamiltonicity and proving its security.

3.1. *Adaptive Instance-Dependent Commitment Schemes*

3.1.1. *Definition*

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

- Com is the bit commitment algorithm: For a bit $b \in \{0,1\}$, an instance $x \in \{0,1\}^*$ and a random string $r \in \{0,1\}^{p(|x|)}$ (where $p(\cdot)$ is a polynomial), $\mathsf{Com}(x, b; r)$ 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 $x \in L$ and $w \in R_x$. For all c and $r \in \{0,1\}^{p(|x|)}$ such that $\mathsf{Com}'(x; r) = c$, and for all $b \in \{0,1\}$, $\mathsf{Adapt}(x, w, c, b, r)$ returns a pair (b, r') such that $c = \mathsf{Com}(x, b; r')$. (In other worlds, Adapt receives a "fake" commitment c and a bit b, and provides an explanation for c as a commitment to the bit b.)

A *decommitment* to a commitment c is a pair (b, r) such that $c = \mathsf{Com}(x, b; r)$.

Note the difference between Com and Com': Com is an ordinary committing algorithm (creating a commitment value for a given bit), while for $x \in L$ algorithm Com' creates commitment values that are not associated to any specific bit. However, given a witness attesting to the fact that $x \in L$, these commitments can later be claimed to be commitments to 0 or to 1 by using algorithm Adapt. We stress that without such a witness, a commitment generated by Com' cannot necessarily be decommitted to any bit.

Security We now define the notion of security for our commitment scheme. Recall that our goal is eventually designing an adaptive zero-knowledge proof and our definition of security is oriented towards this goal.

Let $C_0^x = \{c \mid \exists\, r \text{ s.t. } c = \mathsf{Com}(x, 0; r)\}$ and $C_1^x = \{c \mid \exists\, r \text{ s.t. } c = \mathsf{Com}(x, 1; r)\}$. That is, C_0^x is the set of commitment values that can be decommitted to 0 and C_1^x the set of commitment values that can be decommitted to 1.

Definition 3.2. Let R be an $\mathcal{NP}$ relation and $L = L_R$. We say that $(\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$ is a secure AIDCS for L if the following hold:

1. *Computational hiding*: The ensembles $\{\mathsf{Com}(x, 0)\}_{x\in L}$, $\{\mathsf{Com}(x, 1)\}_{x\in L}$ and $\{\mathsf{Com}'(x)\}_{x\in L}$ are computationally indistinguishable.
2. *Adaptivity*: For all $b \in \{0, 1\}$, the distributions

$$\{(\mathsf{Com}(x, b; U_{p(|x|)}), b, U_{p(|x|)})\}_{x\in L, w\in R_x}$$

and

$$\{(\mathsf{Com}'(x; U_{p(|x|)}), \mathsf{Adapt}(x, w, \mathsf{Com}'(x; U_{p(|x|)}), b, U_{p(|x|)}))\}_{x\in L, w\in R_x}$$

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. *Perfect binding*: For all $x \notin L$, The sets C_0^x and C_1^x are disjoint.

3.1.2. *String Commitments*

We now argue that an adaptive instance-dependent commitment scheme remains secure even when we concatenate bit commitments. First, it is easy to see that the binding and the hiding properties are preserved when concatenating bit commitments. To prove that adaptivity is preserved as well we consider the following experiment between an adversary and a commitment oracle. In this experiment, the adversary outputs a message m (a sequence of bits) and the commitment oracle chooses either to use the committing algorithm Com to commit to each of the bits of the message and output the sequence of the commitments together with the random coins used by Com or to generate fake commitments using algorithm Com', use algorithm Adapt to get random coins that are consistent with the message m and output the fake commitments together with the random coins generated by Adapt. The adversary succeeds in the game if it can distinguish between the choices of the commitment oracle with non-negligible probability. Formally, for a commitment scheme $C = (\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$, an adversary $\mathcal{A}$, an instance $x \in L$, a witness $w \in R_x$ and a value $b \in \{0, 1\}$, consider the following experiment:

The string commitment adaptivity experiment $\mathbf{StrExpAdapt}^b_{\mathcal{A},C}(x, w)$.

1. Upon input $x \in L$ and $w \in R_x$, where $|x| = n$, the adversary $\mathcal{A}$ outputs a message $m \in \{0, 1\}^{q(n)}$.
2. Let $q = q(n)$. If $b = 0$ the commitment $c = \mathsf{Com}(x, m_1; r_1), \ldots, \mathsf{Com}(x, m_q, r_q)$ is computed where $r = r_1, \ldots, r_q$ is uniformly chosen and (c, m, r) is given to $\mathcal{A}$.

   If $b = 1$, the commitment $c = \mathsf{Com}'(x; r_1), \ldots, \mathsf{Com}'(x; r_q)$ is computed where $r = r_1, \ldots, r_q$ is uniformly chosen. Then for all $1 \le i \le q$, $(m_i, r_i') = \mathsf{Adapt}(x, w, \mathsf{Com}'(x; r_i), m_i, r_i)$ is computed. Finally (c, m, r') is given to $\mathcal{A}$, where $r' = r_1', \ldots, r_q'$.
3. $\mathcal{A}$ outputs a value b' and this is the output of the experiment.

The following proposition follows from the adaptivity of the bit commitment scheme and can be proved by a simple hybrid argument:

Proposition 3.3. *Let* $C = (\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$ *be an AIDCS. For every PPT machine* $\mathcal{A}$*, every positive polynomial* $p(\cdot)$ *and all sufficiently large* $x \in L$ *and* $w \in R_x$*,*

$$\left|\Pr\left[\mathsf{StrExpAdapt}^0_{\mathcal{A},C}(x,w)=1\right]-\Pr\left[\mathsf{StrExpAdapt}^1_{\mathcal{A},C}(x,w)=1\right]\right|<\frac{1}{p(|x|)}.$$

3.1.3. *The Construction*

Our construction is almost identical to the trapdoor commitment of [16] (as adapted by [9]), with one small, but crucial difference. We begin by describing the construction of [9]. Let C be a perfectly binding commitment scheme with pseudorandom range and let G be a graph (in [16], G is a Hamiltonian graph generated by the receiver, whereas in [9] it is a Hamiltonian graph that is placed in the common reference string). Then, in order to commit to 0, the committer chooses a random permutation π of the vertices of G and commits to the adjacency matrix of $\pi(G)$ using C. To decommit, it opens all entries and sends π. To commit to 1, the committer chooses a random n-cycle and for all entries in the adjacency matrix corresponding to the edges of the n-cycle, it uses C to commit to 1. In contrast, all other entries are set to a random string (recall that the commitment scheme has a pseudorandom range and thus these values are indistinguishable from commitments using C). To decommit, it opens only the entries corresponding to the edges of the n-cycle. As stated, this scheme is computationally hiding due to the underlying commitment scheme C. In addition, it is computational binding as long as the sender does not know the Hamiltonian cycle in G. We stress that the scheme is not perfectly binding because an all-powerful corrupted committer can find the Hamiltonian cycle in G and send commitments that it can later open to both 0 and 1.

Our key observation is that in the setting of zero-knowledge we can use the graph G*, that is, the statement being proven, as the graph in the above commitment scheme.* This implies that if $G \in HC$, then the commitment scheme is computationally hiding, and if $G \notin HC$ then it is perfectly binding, as required. (As an added bonus, the graph need not be generated by the protocol.) Regarding adaptivity, when $G \in HC$ a commitment to 0 can be opened as a 0 or 1 given a cycle in G.

Formally, we define the commitment scheme $(\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$ as follows:

- $\mathsf{Com}(G, 0)$ chooses a random permutation π of the vertices of G, and sets $H = \pi(G)$. The output of the algorithm is a series of commitments (using the commitment scheme C) to the adjacency matrix of H. Namely, a matrix of size $n \times n$, where the entry (i, j) contains a commitment to 1 if $(i, j) \in E(H)$ and to 0 if $(i, j) \notin E(H)$, where $E(H)$ denotes the set of edges in H.
  $\mathsf{Com}(G, 1)$ chooses a random cycle of size n. The algorithm outputs a matrix of size $n \times n$, where the entries corresponding to the cycle contain a commitment to 1 (using commitment scheme C). All other entries are set to random strings of the same length as the output length of C.
- $\mathsf{Com}'(G)$ acts exactly as $\mathsf{Com}(G, 0)$; that is, $\mathsf{Com}'(G; r) = \mathsf{Com}(G, 0; r)$.

- $\mathsf{Adapt}(G, w, c, 0, r)$: If $\mathsf{Com}'(G; r) \neq c$, then it returns $\perp$. Otherwise, it outputs the bit 0 and r (recall that $\mathsf{Com}'(G; r) = \mathsf{Com}(G, 0; r)$, and therefore $\mathsf{Com}(G, 0; r) = c$).
- $\mathsf{Adapt}(G, w, c, 1, r)$: If $\mathsf{Com}'(G; r) \neq c$, then it returns $\perp$. Otherwise, it outputs the bit 1, the cycle of length n obtained by applying the permutation π (that is a part of r) on the Hamiltonian cycle w, and n^2 strings $\{r_{i,j}\}_{i,j=1}^{n}$ where if the edge $(i, j) \in \pi(w)$, then $r_{i,j}$ is the randomness used by C to commit to 1 and if $(i, j) \notin \pi(w)$, then $r_{i,j}$ is the commitment value that appears in the corresponding entry in c (recall that C has pseudorandom range; therefore these values "look" random).

Proposition 3.4. *Assuming the existence of one way permutations*, $(\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$ *is a secure non-interactive adaptive instance-dependent commitment scheme for the language of Hamiltonicity.*

Proof. We show that $(\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$ fulfills Definition 3.2.

Computational Hiding: The computational hiding of the scheme has been proven in [9], and therefore we omit the proof.

Perfect Binding: Let $G \notin HC$. By the definition, the set C_0^G contains commitments to isomorphic graphs to G and the set C_1^G contains commitments to Hamiltonian cycles. If $G \notin HC$ then a graph cannot be simultaneously isomorphic to G and contain a Hamiltonian cycle, and therefore the sets C_0^G and C_1^G are disjoint.

Adaptivity: We begin by arguing that for all graphs $G \in HC$ and all $w \in R_G$, the distributions $\{(\mathsf{Com}(G, 0; U_{p(n)}), 0, U_{p(n)})\}$ and $\{(\mathsf{Com}'(G; U_{p(n)}), \mathsf{Adapt}(G, w, \mathsf{Com}'(x); U_{p(n)}), 0, U_{p(n)}))\}$ are computationally indistinguishable. This is easy to verify since, as we have mentioned in the construction of our AIDCS, $\mathsf{Com}'(G; U_{p(n)})$ is, in fact, an execution of $\mathsf{Com}(G, 0; U_{p(n)})$ and in this case $\mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 0, U_{p(n)})$ returns a bit 0 and the random coins used by $\mathsf{Com}'(G)$. We conclude that $\{(\mathsf{Com}(G, 0; U_{p(n)}), 0, U_{p(n)})\}$ and $\{(\mathsf{Com}'(G; U_{p(n)}), \mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 0, U_{p(n)}))\}$ are identically distributed.

Now, we argue that for all $G \in HC$ and $w \in R_G$, the distributions $\{(\mathsf{Com}(G, 1; U_{p(n)}), 1, U_{p(n)})\}$ and $\{(\mathsf{Com}'(G; U_{p(n)}), \mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 1, U_{p(n)}))\}$ are computationally indistinguishable. Consider the following experiment: A random $r \in \{0, 1\}^{p(n)}$ is chosen and $c = \mathsf{Com}'(G; r)$ is computed. Then, $(1, r') = \mathsf{Adapt}(G, w, c, 1, r)$ is computed and the output of the experiment is $(\mathsf{Com}(G, 1; r'), 1, r')$. We denote the distribution induced by this experiment by H_{Com}. It is not difficult to see that the distributions $\{(\mathsf{Com}'(G; U_{p(n)}), \mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 1, U_{p(n)}))\}$ and H_{Com} are identically distributed. We will show that $\{(\mathsf{Com}(G, 1; U_{p(n)}), 1, U_{p(n)})\}$ and H_{Com} are computationally indistinguishable. Recall that $\mathsf{Adapt}(G, w, c, 1, r)$ outputs n^2 strings $\{r_{i,j}\}_{i,j=1}^{n}$ where if the edge $(i, j) \in \pi(w)$, then $r_{i,j}$ is the random string used by C to commit to 1 and if $(i, j) \notin \pi(w)$, then $r_{i,j}$ is the value that appears in the corresponding entry in c. That is, if $(i, j) \in \pi(w)$, the corresponding string $r_{i,j}$ is a truly random string and if $(i, j) \notin \pi(w)$, then the corresponding string $r_{i,j}$ is an output of the commitment

scheme C. We have taken C to have a pseudorandom range, and this implies that the distribution $\{\mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 1, U_{p(n)})\}$ is computationally indistinguishable from $\{U_{p(n)}\}$.

Now, assume that $\{(\mathsf{Com}(G, 1; U_{p(n)}), 1, U_{p(n)})\}$ and H_{Com} are not computationally indistinguishable. Informally speaking, there exists a PPT machine D that can distinguish between $\{(\mathsf{Com}(G, 1; U_{p(n)}), 1, U_{p(n)})\}$ and H_{Com} with a non-negligible probability. We construct a distinguisher D' that can distinguish $\{\mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 1, U_{p(n)})\}$ and $\{U_{p(n)}\}$. D' gets an input an r that is distributed according to $\{U_{p(n)}\}$ or $\{\mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 1, U_{p(n)})\}$. D' then computes $c = \mathsf{Com}(G, 1, r)$, hands $(c, 1, r)$ to D and outputs the output of D. Now, if r is distributed according to $\{U_{p(n)}\}$, then the input of D is distributed according to $\{(\mathsf{Com}(G, 1; U_{p(n)}), 1, U_{p(n)})\}$. On the other hand, if r is distributed according to $\{\mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 1, U_{p(n)})\}$, then the input of D is distributed as H_{Com}, and therefore D' distinguishes $\{\mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 1, U_{p(n)})\}$ and $\{U_{p(n)}\}$ with a non-negligible probability, contradicting the computational indistinguishability of $\{\mathsf{Adapt}(G, w, \mathsf{Com}'(x; U_{p(n)}), 1, U_{p(n)})\}$ and $\{U_{p(n)}\}$. □

Remark (One-way functions). Note that we can reduce our computational assumptions to the existence of one-way functions (rather than one-way permutations) by replacing the commitment scheme C by the scheme of [35] (which has a pseudorandom range as well). The resulting scheme will be an *interactive* Adaptive Instance Dependent Commitment Scheme with computational hiding and statistical binding. Also note that all proofs can be extended to the case of non-uniform adversaries assuming the existence of one-way functions that are hard to invert for non-uniform adversaries.

3.2. *Adaptive Zero Knowledge Proofs for All $\mathcal{NP}$*

In this section, we show that the proof system for Hamiltonicity presented in [5] is an adaptive zero-knowledge proof system when the ordinary commitment scheme is replaced by an adaptive instance-dependent commitment scheme.

Theorem 3.5. *If there exists an AIDCS for an $\mathcal{NP}$-complete problem with computational hiding and adaptivity against non-uniform adversaries, then every NP language has an adaptive zero knowledge proof.*

Proof. We present an adaptive zero knowledge proof for Hamiltonicity; the generalization to any language in $\mathcal{NP}$ is achieved by reducing the language to Hamiltonicity. Let $C = (\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$ be an AIDCS for Hamiltonicity.

Protocol 1 (Adaptive zero-knowledge proof for HC).

- *The verifier's input:* A graph $G = (V, E)$ where $|V| = n$.
- *The prover's input:* A graph $G = (V, E)$ where $|V| = n$ and a Hamiltonian cycle w in G.
- *The protocol:*

  1. The prover chooses a random permutation π of the vertices and commits to the adjacency matrix of $\pi(G)$ using algorithm Com.

2. The verifier sends a random bit σ to the prover.
3. If $\sigma = 0$ the prover sends π to the verifier and decommits to all entries of the adjacency matrix.
   If $\sigma = 1$, the prover decommits to the entries of the Hamiltonian cycle in $\pi(G)$.
4. If $\sigma = 0$ the verifier checks that the decommitted graph is indeed $\pi(G)$.
   If $\sigma = 1$ the verifier checks that the decommitted entries form a Hamiltonian cycle of size n.

Claim 3.6. *Protocol 1 is an efficient-prover interactive proof for the language of Hamiltonicity.*

Proof. It is easy to see that the strategy of the verifier can be implemented in probabilistic polynomial-time and that, given a Hamiltonian cycle in G, the strategy of the prover can be implemented in probabilistic polynomial-time as well.

Proving the completeness property of the protocol is straightforward. We will prove soundness of $\frac{1}{2}$. Let $G \notin HC$, and assume there exists a prover strategy P^* such that $\Pr[\langle P^*, V\rangle(G) = 1] > \frac{1}{2}$. Let p_i be P^* ith message to V. Assuming P^* convinces V with probability that is greater than $\frac{1}{2}$, there exist messages p'_1 and p_2^0 and p_2^1 such that both $V(p'_1, 0, p_2^0)$ and $V(p'_1, 1, p_2^1)$ accept. Namely, p_2^0 is a decommitment of p'_1 to an isomorphic graph of G and p_2^1 is a decommitment of some entries in the adjacency matrix that is represented by p'_1 that form a Hamiltonian cycle. Since the commitment scheme C is perfectly binding when $G \notin HC$, this implies that G contains a Hamiltonian cycle, and that is a contradiction to $G \notin HC$. □

Claim 3.7. *Protocol 1 is secure against adaptive adversaries.*

Proof. Let $\mathcal{A}$ be a PPT adaptive adversary that interacts with the prover and the verifier in the real-life run of the protocol and let $\mathcal{Z}$ be a PPT environment. We construct an ideal-process adversary (simulator) $\mathcal{SIM}$ such that

$$\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \stackrel{c}{\equiv} \text{IDEAL}^{ZK}_{L,\mathcal{SIM},\mathcal{Z}}.$$

For the sake of simplicity, we begin by describing a simulator $\mathcal{SIM}$ whose running time might not be expected polynomial time and then we show how this simulator can be modified, using ideas from [22], to an expected polynomial time simulator $\mathcal{SIM}'$ such that

$$\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \stackrel{c}{\equiv} \text{IDEAL}^{ZK}_{L,\mathcal{SIM}',\mathcal{Z}}.$$

$\mathcal{SIM}$ starts with an input G and a random tape $r_{\mathcal{SIM}}$. $\mathcal{SIM}$ invokes a simulated copy of $\mathcal{A}$ on a uniform random tape. $\mathcal{SIM}$ simulates the interaction of $\mathcal{A}$ with the environment $\mathcal{Z}$ as follows: Every input $\mathcal{SIM}$ receives from the environment $\mathcal{Z}$ is written on the input tape of $\mathcal{A}$, as if it came from $\mathcal{A}$'s environment. Every output written by $\mathcal{A}$ on its output tape is written by $\mathcal{SIM}$ on its own output tape. If $\mathcal{SIM}$ receives some auxiliary input from $\mathcal{Z}$ at the outset, then $\mathcal{SIM}$ hands it to $\mathcal{A}$. To simplify our analysis, we

divide our protocol into 3 rounds: The prover's first message, the verifier's first message and the prover's second message, where the adversary might corrupt any of the parties at the onset of each round or in the postexecution corruption phase.

Our description of $\mathcal{SIM}$ consists of 2 parts: simulating corruptions of the parties and simulating the run of the protocol.

- *Simulating the run of the protocol:* We divide the description of the simulation into cases as following.

  1. *The prover is uncorrupted in the first round.* $\mathcal{SIM}$ generates the prover's first message as follows. $\mathcal{SIM}$ chooses a random bit $b \in \{0, 1\}$. If $b = 0$, it chooses a random permutation π and uses algorithm Com to commit to the adjacency matrix of $\pi(G)$.
     If $b = 1$, $\mathcal{SIM}$ creates an adjacency matrix of a graph containing only a random cycle of length n (all other entries are set to 0). It then uses Com to commit to all 1's in the adjacency matrix. All other entries are filled with commitment values created by algorithm Com'.
     We proceed to the second round.
     (a) *The verifier is corrupted in the second round.* If $\mathcal{A}$ aborts, $\mathcal{SIM}$ aborts as well. Otherwise, $\mathcal{A}$ generates a bit σ and $\mathcal{SIM}$ sets σ as the verifier's message. We move to the third round.
        (i) *The prover is uncorrupted in the third round.* $\mathcal{SIM}$ generates the prover's second message as follows: If $b = \sigma = 0$ then $\mathcal{SIM}$ sets the prover's second message to be decommitments to all entries at the adjacency matrix and π. If $b = \sigma = 1$, $\mathcal{SIM}$ decommits to all 1's in the adjacency matrix (and since these entries were created using Com, $\mathcal{SIM}$ can decommit). In both cases, the run of the protocol terminates. The corrupted verifier's output is a special symbol $\perp$ and the output of the prover is the empty string λ. $\mathcal{SIM}$ outputs the simulated adversary's internal state. A postexecution corruption step starts and at the end $\mathcal{Z}$ halts with an output.
        If $b \neq \sigma$, $\mathcal{SIM}$ rewinds to the beginning of the first corruption stage (that is, the outset of the first round) and proceeds as above. In this case, if the corruptions made by $\mathcal{A}$ do not lead $\mathcal{SIM}$ to the same scenario (that is, the verifier is corrupted before the second round and the prover is uncorrupted in the third round), $\mathcal{SIM}$ keeps rewinding to the outset of the first round until $\mathcal{SIM}$ ends up in the same scenario as the current one. We also note that if $\mathcal{A}$ aborts before sending the verifier's message or if again $b \neq \sigma$, $\mathcal{SIM}$ has to rewind again.
        (ii) *The prover is corrupted in the third round.* If the simulated copy of $\mathcal{A}$ aborts, then $\mathcal{SIM}$ aborts. Otherwise, $\mathcal{A}$ generates the prover's second message. The output of the corrupted parties is the special symbol $\perp$ and $\mathcal{SIM}$ outputs the simulated adversary's internal state. Since both parties are corrupted, no postexecution corruption step takes place and $\mathcal{Z}$ halts with an output.
     (b) *The verifier is uncorrupted in the second round.* In this case, $\mathcal{SIM}$ sets $\sigma = b$ to be the verifier's first message (where b is as chosen by $\mathcal{SIM}$ above).

(i) *The prover is uncorrupted in the third round.* Since $\sigma = b$, the simulator simply decommits appropriately to the commitment it sent, and the third round terminates. At the end of the protocol, $\mathcal{SIM}$ outputs the simulated adversary's internal state. A postexecution corruption stage starts and at the end $\mathcal{Z}$ halts with an output.

(ii) *The prover is corrupted in the third round.* If the simulated copy of $\mathcal{A}$ aborts, $\mathcal{SIM}$ aborts. Otherwise, $\mathcal{A}$ generates the prover's second message and the protocol ends. $\mathcal{SIM}$ outputs the simulated adversary's internal state. If the verifier is uncorrupted, a postexecution corruption stage starts and at the end $\mathcal{Z}$ halts with an output.

2. *The prover is corrupted in the first round.* If the simulated copy of $\mathcal{A}$ aborts, then $\mathcal{SIM}$ aborts. Otherwise, $\mathcal{A}$ generates the prover's first message and the first round ends.
   (a) *The verifier is corrupted in the second round.* If the simulated copy of $\mathcal{A}$ aborts, then $\mathcal{SIM}$ aborts. Otherwise, $\mathcal{A}$ generates the verifier's first message and the prover's second message. At the end, both parties output the special symbol $\perp$ and $\mathcal{SIM}$ outputs the simulated adversary's internal state. Since both parties are corrupted, no postexecution corruption stage has taken place and $\mathcal{Z}$ halts with an output.
   (b) *The verifier is uncorrupted in the second round.* $\mathcal{SIM}$ chooses a random bit σ and sets it as the verifier's first message. The simulated copy of $\mathcal{A}$ generates the prover's second message (or aborts) and the protocol ends. $\mathcal{SIM}$ outputs the simulated adversary's internal state. If the verifier is uncorrupted, a postexecution corruption stage starts and at the end $\mathcal{Z}$ halts with an output.

- *Simulating Corruptions:*

1. *$\mathcal{A}$ corrupts the verifier at the outset of the first or second rounds:* $\mathcal{SIM}$ corrupts the verifier and hands a truly random string to $\mathcal{A}$ as the verifier's random tape r_V.
2. *$\mathcal{A}$ corrupts the verifier at the outset of the third round or in the postexecution corruption step:* In this case, the verifier's first message has been set by $\mathcal{SIM}$ to a random bit σ. $\mathcal{SIM}$ corrupts the verifier and provides $\mathcal{A}$ with a random tape r_V that is consistent with σ.
3. *$\mathcal{A}$ corrupts the prover at the outset of the first round:* $\mathcal{SIM}$ corrupts the prover, learns its input tape that includes the witness w, sets r_P to be a truly random string and provides $\mathcal{A}$ with w and r_P.
4. *$\mathcal{A}$ corrupts the prover at the outset of the second or third round or at the postexecution corruption step:* $\mathcal{SIM}$ corrupts the prover in the ideal-process run of the protocol and learns the witness w which is a part of its input tape. At this stage, $\mathcal{A}$ expects to learn the input and the random tape of P, and $\mathcal{SIM}$ has to generate a random tape for P that is consistent with the prover's first message. If $b = 0$, the prover's first message has been created exactly as in a real run of the protocol and the simulator can provide $\mathcal{A}$ with the random coins used to create the commitment. If $b = 1$, the prover's first message is a commitment to a graph that contains a single cycle of length n; we denote this

graph by C_n. The simulator finds an isomorphism π between the Hamiltonian cycle w in G and between C_n. It then computes $H = \pi(G)$. For every edge $(u, v) \in C_n$, $\mathcal{SIM}$ provides $\mathcal{A}$ with the randomness used by Com to commit to 1. For every edge $(u, v) \in H$ and $(u, v) \notin C_n$, $\mathcal{SIM}$ uses algorithm Adapt to obtain random coins such that the appropriate commitment value in the adjacency matrix is a commitment to 1. For every $(u, v) \notin H$, $\mathcal{SIM}$ uses algorithm Adapt to get random coins such that the commitment value in the adjacency matrix is a commitment to 0. $\mathcal{SIM}$ provides the simulated copy of $\mathcal{A}$ with the input w and the set of random coins described above as well as with π.

To see that the running time of $\mathcal{SIM}$ might not be expected polynomial time, consider the case that with probability α the simulator $\mathcal{SIM}$ reaches case 1a, and yet $b \neq \sigma$. If this happens, then $\mathcal{SIM}$ must rewind until it reaches case 1a again, but this time with $b = \sigma$. Letting β be the probability that this happens, we have that the expected running time of $\mathcal{SIM}$ is $\alpha \cdot \frac{1}{\beta}$. Now, it is not difficult to show that α and β are negligibly close; otherwise, we can use $\mathcal{A}$ and $\mathcal{SIM}$ to break the hiding property of the commitments (observe that the commitments sent by $\mathcal{SIM}$ when $b = 0$ differ from those when sent by $\mathcal{SIM}$ when $b = 1$). However, even though α and β are negligibly close, this does not guarantee that α/β is polynomial; in particular, if $\alpha = 2^{-n}$ and $\beta = 2^{-2n}$ then $\alpha/\beta = 2^n$. We use the methodology of [22] in order to solve this problem and ensure that the expected running-time of the simulator is polynomial. Specifically, we modify the simulator $\mathcal{SIM}$ to a simulator $\mathcal{SIM}'$ so that, whenever the simulation reaches case 1a, it first computes an estimate $\tilde{\alpha}$ of α by repeating the simulation steps 1 and 2 until this case (1a and $b \neq \sigma$) occurs a fixed polynomial number of times; denote this number by $q(n)$. By taking a sufficiently large polynomial $q(\cdot)$, we can make sure that with probability $1 - 2^{-n^2}$ the estimate $\tilde{\alpha}$ is within a constant factor of α (a full proof of this fact appears in [25, Sect. 6.5.3]). Following this, the modified simulator $\mathcal{SIM}'$ rewinds the adversary to the outset of the first corruption stage and makes at most $n/\tilde{\alpha}$ attempts to reach case 1a with $b = \sigma$. If $\mathcal{SIM}'$ does not reach this scenario within $n/\tilde{\alpha}$ tries, then it outputs Time-Out and halts. In addition to the above, $\mathcal{SIM}$ counts the number of executions of $\mathcal{A}$ and halts outputting Time-Out if this number exceeds 2^n.

We now show that the expected running time of $\mathcal{SIM}'$ is polynomial. We use the enumeration of the cases above (in the simulation of the run of the protocol) to indicate which parties have been corrupted and when in a specific run of the simulation. For example, when we refer to case 2, we refer to the case where the prover is corrupted at the outset of the first run. Similarly, when we refer to case 1a, we refer to the case where the prover is uncorrupted at the outset of the first round and the verifier is corrupted at the outset of the second round and when we refer to case 1a, we refer to the case where the prover is uncorrupted at the outset of the first round, the verifier is corrupted at the outset of the second round and the prover is corrupted at the outset of the third round. It is easy to see that a single iteration of $\mathcal{SIM}'$ (where no rewinding is carried out) takes a polynomial number of steps (recall that the real-life adversary $\mathcal{A}$ is a PPT machine and therefore invoking a simulated copy of $\mathcal{A}$ takes polynomial time) and that $\mathcal{SIM}'$ rewinds only if it reaches case 1a in the simulation (when the prover is uncorrupted at the third round and the verifier is corrupted before the second round) and its random bit

b does not equal the verifier's first message σ. We denote by $\hat{b}$ the random bit of $\mathcal{SIM}'$ in its first iteration and by $\hat{\sigma}$ the verifier's message in the first iteration. Denoting by TIME the random variable of $\mathcal{SIM}'$'s running time, we have:

$$\begin{aligned} E[\text{TIME}] &= E[\text{TIME} \mid \text{case 1a} \wedge \hat{b} \neq \hat{\sigma}] \cdot \Pr[\text{case 1a} \wedge \hat{b} \neq \hat{\sigma}] \\ &\quad + E\left[\text{TIME} \mid \neg(\text{case 1a} \wedge \hat{b} \neq \hat{\sigma})\right] \cdot \Pr\left[\neg(\text{case 1a} \wedge \hat{b} \neq \hat{\sigma})\right] \\ &\leq E[\text{TIME} \mid \text{case 1a} \wedge \hat{b} \neq \hat{\sigma}] \cdot \Pr[\text{case 1a} \wedge \hat{b} \neq \hat{\sigma}] + \text{poly}(\mathsf{n}). \end{aligned}$$

We now show that

$$E[\text{TIME} \mid \text{case 1a} \wedge \hat{b} \neq \hat{\sigma}] \cdot \Pr[\text{case 1a} \wedge \hat{b} \neq \hat{\sigma}]$$

is polynomial.

Let $\alpha = \Pr[\text{case 1a} \wedge \hat{b} \neq \hat{\sigma}]$. Recall that whenever $\mathcal{SIM}'$ reaches case 1a and $b \neq \sigma$, it repeats the simulation until $q(n)$ occurrences of this case occurs, obtains an estimate $\tilde{\alpha}$ of α, and then rewinds the adversary for an expected $n/\tilde{\alpha}$ number of times. If during these $n/\tilde{\alpha}$ rewindings, the adversary reaches case 1a with $b = \sigma$, then it can complete the simulation, and otherwise it outputs $\perp$. We distinguish between two cases. In the first case, $\tilde{\alpha}$ is not within a constant factor of α. When this happens, we do not know anything about the value $\tilde{\alpha}$ and so the only thing that we can say is that the running time of $\mathcal{SIM}'$ is strictly bounded by $2^n \cdot \text{poly}(n)$ (since $\mathcal{SIM}'$ does not carry out more that 2^n executions of $\mathcal{A}$). However, this case happens with probability at most 2^{-n^2}, and so this adds only a negligible factor to the expected running time of $\mathcal{SIM}'$. In the second case, $\tilde{\alpha}$ is within a constant factor of α and so $\tilde{\alpha} = c \cdot \alpha$ for some constant c. Thus, $\mathcal{SIM}'$'s expected running-time in this case is $\frac{q(n)}{\alpha} \cdot \text{poly}(n)$ steps (to find $\tilde{\alpha}$) plus $\frac{n}{\tilde{\alpha}} \cdot \text{poly}(n) = \frac{n}{c \cdot \alpha} \cdot \text{poly}(n)$ steps (rewinding $\mathcal{A}$). We conclude that

$$\begin{aligned} &E[\text{TIME} \mid \text{case 1a} \wedge \hat{b} \neq \hat{\sigma}] \cdot \Pr[\text{case 1a} \wedge \hat{b} \neq \hat{\sigma}] \\ &\leq \text{poly}(n) \cdot \left(2^{-n^2} \cdot 2^n + \frac{q(n)}{\alpha} + \frac{n}{c \cdot \alpha}\right) \cdot \alpha \leq \text{poly}(n) \cdot \left(1 + q(n) + \frac{n}{c}\right), \end{aligned}$$

which is polynomial. We have therefore proven the following proposition:

Proposition 3.8. *The expected running time of $\mathcal{SIM}'$ is polynomial in n.*

We now need to show that the output distribution of $\mathcal{Z}$ in the real-process is computationally indistinguishable from the output of $\mathcal{Z}$ in the ideal-process:

$$\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \stackrel{\text{c}}{\equiv} \text{IDEAL}^{ZK}_{L,\mathcal{SIM}',\mathcal{Z}}.$$

The proof here is identical to that of [22], and we therefore just repeat the main ideas and omit the full proof. As in [22], we can reduce the analysis to analyzing the output distribution of the original simulator $\mathcal{SIM}$. The only difference between $\mathcal{SIM}$ and $\mathcal{SIM}'$ is in the cases where $\mathcal{SIM}'$ outputs Time-Out. However, similarly to [22], it can be shown that the probability that $\mathcal{SIM}'$ outputs Time-Out is negligible.

Claim 3.9. *The probability that $\mathcal{SIM}'$ outputs* Time-Out *is a negligible function of n.*

Proof. Let $\Delta(x,r)$ denote the event that $\mathcal{SIM}'$ on input x and random tape r outputs Time-Out. $\mathcal{SIM}'$ outputs Time-Out in the following cases:

1. It invokes $\mathcal{A}$ more than 2^n times: We have already shown that the expected running time of $\mathcal{SIM}'$ is polynomial, and thus by Markov's inequality, this event happens with at most negligible probability; we bound this probability by $2^{-n/2}$.
2. After $n/\tilde{\alpha}$ attempts to rewind, $\mathcal{SIM}'$ does not obtain an execution in which case 1a occurs with $b=\sigma$: Let β denote the probability that $\mathcal{SIM}'$ reaches case 1a and $b=\sigma$. $\mathcal{SIM}'$ outputs Time-Out if in all $n/\tilde{\alpha}$ tries, this event does not happen.

We therefore bound the probability that $\Delta(x,r)$ occurs as follows:

$$\begin{aligned}\Pr[\Delta(x,r)] &= 2^{-n/2} + \alpha \cdot \Pr\left[\frac{n}{\tilde{\alpha}} \text{ failures}\right] \\ &= 2^{-n/2} + \alpha \cdot \sum_{i\geq 1} \Pr\left[\lfloor 1/\tilde{\alpha}\rfloor = i\right] \cdot (1-\beta)^{i\cdot n} \\ &< 2^{-n/2} + \alpha \cdot \left(\Pr\left[\frac{\alpha}{\tilde{\alpha}} = \Theta(1)\right] \cdot (1-\beta)^{n^2/\alpha} + \Pr\left[\frac{\alpha}{\tilde{\alpha}} \neq \Theta(1)\right]\right).\end{aligned}$$

Recall that $\Pr[\frac{\alpha}{\tilde{\alpha}} \neq \Theta(1)]$ is negligible in n. We bound this probability with $2^{-n/2}$ and obtain that

$$\begin{aligned}\Pr[\Delta(x,r)] &< 2^{-n/2} + \alpha \cdot \left(\Pr\left[\frac{\alpha}{\tilde{\alpha}} = \Theta(1)\right] \cdot (1-\beta)^{n^2/\alpha} + 2^{-n/2}\right) \\ &< 2 \cdot 2^{-n/2} + \alpha \cdot (1-\beta)^{n^2/\alpha}.\end{aligned}$$

We now show that $\alpha \cdot (1-\beta)^{n^2/\alpha}$ is negligible. Assume by contradiction that there exist a polynomial $p(\cdot)$ and an infinite sequence of $\{x_n\}$ (with $|x|=n$) and an infinite sequence of random tapes $\{r_n\}$ such that $\alpha \cdot (1-\beta)^{n^2/\alpha} > \frac{1}{p(n)}$. It holds that for all these n's, $\alpha > \frac{1}{p(n)}$. Similarly as in [22], we divide our analysis into two cases:

1. For infinitely many of these n's, $\beta \geq \frac{\alpha}{2}$. For these n's, it holds that

$$\alpha \cdot (1-\beta)^{n^2/\alpha} \leq (1-\beta)^{n^2/\alpha} \leq \left(1-\frac{\alpha}{2}\right)^{n^2/\alpha} = \left(\left(1-\frac{\alpha}{2}\right)^{2/\alpha}\right)^{n^2/2} < e^{-n^2/2},$$

   which contradicts our assumption that $\alpha \cdot (1-\beta)^{n^2/\alpha} > \frac{1}{p(n)}$.
2. For infinitely many of these n's, $\beta < \frac{\alpha}{2}$. For these n's, it holds that $|\alpha - \beta| > \frac{\alpha}{2} > \frac{1}{2p(n)}$. Recall that $\alpha = \Pr[\text{case 1a} \wedge b \neq \sigma]$ and $\beta = \Pr[\text{case 1a} \wedge b = \sigma]$. Therefore,

$$\left|\Pr[\text{case 1a} \wedge b \neq \sigma] - \Pr[\text{case 1a} \wedge b = \sigma]\right| > \frac{1}{2p(n)}.$$

However,

$$\begin{aligned}&\left|\Pr[\text{case 1a} \wedge b \neq \sigma] - \Pr[\text{case 1a} \wedge b = \sigma]\right| \\ &\quad = \left|\Pr[\text{case 1a}] \cdot \Pr[b \neq \sigma \mid \text{case 1a}] - \Pr[\text{case 1a}] \cdot \Pr[b = \sigma \mid \text{case 1a}]\right| \\ &\quad = \Pr[\text{case 1a}] \cdot \left|\Pr[b \neq \sigma \mid \text{case 1a}] - \Pr[b = \sigma \mid \text{case 1a}]\right| > \frac{1}{2p(n)}.\end{aligned}$$

This implies that for these n's both

$$\Pr[\text{case 1a}] > \frac{1}{2p(n)}$$

and

$$\left|\Pr[b \neq \sigma \mid \text{case 1a}] - \Pr[b = \sigma \mid \text{case 1a}]\right| > \frac{1}{2p(n)},$$

and now we can use standard arguments to show that this contradicts the computational hiding of the commitment scheme. □

Since $\mathcal{SIM}$ and $\mathcal{SIM}'$ output the same distribution when $\mathcal{SIM}'$ does not output Time-Out, we have that it suffices to consider the original simulator $\mathcal{SIM}$ and show that

$$\text{REAL}_{\Pi,\mathcal{A},\mathcal{Z}} \stackrel{c}{\equiv} \text{IDEAL}^{ZK}_{L,\mathcal{SIM},\mathcal{Z}}.$$

The standard way for showing that the two ensembles are computationally indistinguishable is to show that if the ensembles can be distinguished, then we can break the hiding property of the commitment scheme. This is demonstrated by using the simulator algorithm to construct a PPT machine that can distinguish commitments to different values. However, when trying to apply this argument to our case, we encounter a difficulty because the running time of $\mathcal{SIM}$ is not strictly polynomial. In fact, since we consider $\mathcal{SIM}$ here and not $\mathcal{SIM}'$, it is not even guaranteed to be expected polynomial-time. Therefore, a distinguisher for the commitment scheme cannot use $\mathcal{SIM}$ as a subroutine. In [22], this problem is solved by truncating the executions of $\mathcal{SIM}$ that are too long. Specifically, assume that the distributions are distinguished by a PPT distinguisher D with probability $\epsilon(n)$ where $\epsilon \geq \frac{1}{p(n)}$ for a polynomial $p(\cdot)$ and infinitely many n's. Let S be set of n's for which this holds. We consider two cases:

1. For infinitely many n's in S, it holds that the probability that the corruptions are made as in case 1a is at least $\epsilon(n)/3 = \frac{1}{3p(n)}$. Using standard arguments, we can show that if $\Pr[\text{case 1a}] \geq \frac{1}{3p(n)}$, then it must hold that $\Pr[b = \sigma \mid \text{case 1a}] \geq \frac{1}{4}$, and therefore $\Pr[\text{case 1a} \wedge b = \sigma] \geq \frac{1}{12p(n)}$. In this case, the expected number of times that $\mathcal{SIM}$ has to rewind the adversary until it reaches case 1a and $b = \sigma$ is $12p(n)$, and therefore the probability that more than $24p^2(n)$ rewindings are made is at most $\frac{1}{2p(n)} = \epsilon(n)/2$. Thus, if we truncate the run of $\mathcal{SIM}$ after $24p^2(n)$ rewindings, the output is at most $\epsilon(n)/2$ from the output of $\mathcal{SIM}$. Therefore,

D can distinguish the output of the truncated $\mathcal{SIM}$ from the real execution with gap at least $\epsilon/2$, in contradiction to the computational hiding of the commitment scheme.

2. For infinitely many n's in S, it holds that the probability that the corruptions in the real world are made as in case 1a is less than $\epsilon(n)/3 = \frac{1}{3p(n)}$. We consider two subcases:
   (a) In the ideal world, the probability that corruptions are made as in case 1a is at least $\epsilon(n)/2$. In this case, there is a gap of at least $\epsilon(n)/6$ between the probability that the adversary's corruptions strategy is as in case 1a in the real world and the ideal world. In this case, we use the adversary to distinguish commitments made in the real world and commitments made by $\mathcal{SIM}$ in contradiction to the computational hiding of the commitment scheme.
   (b) In the ideal world, the probability that corruptions are made as in case 1a is at most $\epsilon(n)/2$. This means that both in the real and the ideal worlds, with probability at least $1 - \epsilon(n)/2$, the adversary's corruptions strategy does not lead it to case 1a. D must distinguish between executions in the real word where corruptions are not made as in case 1a and executions in the ideal world where corruptions are not made as in case 1a with gap at least $\epsilon(n)/2$. We note that in these executions the running time of $\mathcal{SIM}$ is polynomial, and therefore we can use D and $\mathcal{SIM}$ to distinguish between commitments made in the real world and commitments made by the simulator, contradicting the computational hiding of the commitment scheme. □

This concludes the proof of Theorem 3.5. □

4. Adaptive Oblivious Transfer

In this section, we prove a black-box separation of adaptively secure oblivious transfer from enhanced trapdoor permutations. We prove our black-box separation in the following steps. First, in Sect. 4.1 we define Γ- and Δ-oracles, where a Γ-oracle essentially represents an enhanced trapdoor permutation and a Δ-oracle represents a type of symmetric encryption scheme. In Sect. 4.2, we show that enhanced trapdoor permutations exist relative to most Γ-oracles. Then, in Sect. 4.3 we show that if there exists a protocol for securely computing any functionality in the presence of *adaptive* adversaries relative to Γ-oracles, then there exists a protocol for securely computing the same functionality in the presence of *static* adversaries relative to Δ-oracles. The next step of the proof is to then show that if $\mathcal{P} = \mathcal{NP}$, then for measure 1 of random Δ-oracles no statically secure OT_1^2 exists (we use OT_1^2 as a shorthand for 1-out-of-2 oblivious transfer). This is done by using the original black-box separation of key agreement from one-way functions [27], and the fact that key agreement can be obtained from statically secure oblivious transfer; see Sect. 4.4. We conclude that if $\mathcal{P} = \mathcal{NP}$, then for measure 1 of random Γ-oracles no adaptively secure OT_1^2 exists (see Sect. 4.5).

4.1. *Oracle Definitions*

We begin by defining (asymmetric) Γ- and (symmetric) Δ-oracles which are used in our proof.

Γ-Oracles Informally speaking, a Γ-oracle is supposed to model an enhanced trapdoor permutation. Thus, it has an oracle for specifying a function and its trapdoor, and an oracle for computing the function (given the function identifier) and inverting it (given the trapdoor). The functions themselves are over all of $\{0,1\}^n$ and thus it is trivial to sample an element without knowing its inverse (as is required for enhanced trapdoor permutations). Formally, we define a Γ-oracle to be an oracle containing the following functions:

- $G_\Gamma(\cdot)$ is an injective function that for every $tid \in \{0,1\}^n$ returns an $fid \in \{0,1\}^{2n}$.
- A function $F(\cdot,\cdot)$, such that for every $fid \in Range(G_\Gamma)$, $F(fid,\cdot)$ is a permutation over $\{0,1\}^n$ and for every $fid \notin Range(G_\Gamma)$ and every $x \in \{0,1\}^n$, $F(fid,x) = \perp$.
- A function F^{-1} satisfying $F^{-1}(tid, F(fid,x)) = x$ for every $x \in \{0,1\}^n$ and every fid, tid such that $G_\Gamma(tid) = fid$.

Uniform Distribution over Oracles—Notation We denote by $\mathcal{U}_\Gamma$ the uniform distribution over Γ-oracles. Namely, an oracle $O_\Gamma = (G_\Gamma, F, F^{-1})$ is distributed according to $\mathcal{U}_\Gamma$ if G_Γ is a uniformly distributed injective function from $\{0,1\}^n$ to $\{0,1\}^{2n}$ and for every $fid \in Range(G_\Gamma)$, $F(fid,\cdot)$ is a uniformly distributed permutation over $\{0,1\}^n$. We write "O_Γ is a random Γ-oracle" as shorthand for "O_Γ is distributed according to $\mathcal{U}_\Gamma$".

Δ-Oracles Informally, a Δ-oracle is a symmetric oracle, meaning that anyone with the ability to compute the function also has the ability to invert it. Specifically, we define a function P and its inverse that is analogous to F and F^{-1} in a Γ-oracle. Formally, we define a "Δ-oracle" to be an oracle containing the following functions:

- G_Δ is an injective function from $\{0,1\}^n$ to $\{0,1\}^{2n}$.
- A function $P(\cdot,\cdot)$ such that for every $fid \in Range(G_\Delta)$, $P(fid,\cdot)$ is a permutation over $\{0,1\}^n$. For $fid \notin Range(G_\Delta)$ and every $x \in \{0,1\}^n$, $P(fid,x) = \perp$.
- P^{-1} is the inversion algorithm of P. Namely for every $fid \in Range(G_\Delta)$ and $x \in \{0,1\}^n$, $P^{-1}(fid, P(fid,x)) = x$. For $fid \notin Range(G_\Delta)$ and every $x \in \{0,1\}^n$, $P^{-1}(fid,x) = \perp$.

We denote by $\mathcal{U}_\Delta$ the uniform distribution over Δ-oracles: The oracle $O_\Delta = (G_\Delta, P, P^{-1})$ is distributed according to $\mathcal{U}_\Delta$, if G_Δ is a uniformly distributed injective function from $\{0,1\}^n$ to $\{0,1\}^{2n}$ and for every $fid \in Range(G_\Delta)$, $P(fid,\cdot)$ is a uniformly distributed permutation over $\{0,1\}^n$. We sometimes write "O_Δ is a random Δ-oracle" instead of "O_Δ is distributed according to $\mathcal{U}_\Delta$".

Note the difference between Γ-oracles and Δ-oracles. Γ-oracles have an asymmetric nature: F and its inversion oracle F^{-1} use different keys. On the contrary, Δ-oracles have a symmetric nature: identical keys are used by P and its inversion oracle P^{-1}. (For this reason, we used a "symmetric" character Δ for Δ-oracles and an "asymmetric" character Γ for Γ-oracles.)

Γ-Oracles Versus Δ-Oracles We now show a bijection mapping ϕ that maps every Γ-oracle to a corresponding Δ-oracle. Let $O_\Gamma = (G_\Gamma, F, F^{-1})$ be a Γ-oracle. $\phi(O_\Gamma)$ is the tuple of functions (G_Δ, P, P^{-1}) satisfying:

- For every $tid \in \{0,1\}^n$, we define $G_\Delta(tid) = G_\Gamma(tid)$.

- For every $fid \in \{0,1\}^{2n}$ and $x \in \{0,1\}^n$, we define $P(fid, x) = F(fid, x)$.
- P^{-1} is the inversion algorithm of P.

Claim 4.1. *The mapping ϕ is a bijection from the set of Γ-oracles to the set of Δ-oracles.*

Proof. We first show that for every O_Γ, $\phi(O_\Gamma) = (G_\Delta, P, P^{-1})$ is indeed a Δ-oracle:

- G_Δ is an injective function because G_Γ is an injective function.
- For $fid \in Range(G_\Delta)$ it holds that $fid \in Range(G_\Gamma)$ and since $F(fid, \cdot)$ is a permutation over $\{0,1\}^n$, $P(fid, \cdot)$ is a permutation over $\{0,1\}^n$ as well.
- For $fid \notin Range(G_\Delta)$ it holds that $fid \notin Range(G_\Gamma)$. For every $x \in \{0,1\}^n$, it holds that $F(fid, x) = \perp$ and thus $P(fid, x) = \perp$ as desired.

We show that the mapping is injective. Let (G_Γ, F, F^{-1}) and $(\widehat{G}_\Gamma, \widehat{F}, \widehat{F}^{-1})$ be two different Γ-oracles. That is, there exists a tid such $G_\Gamma(tid) \neq \widehat{G}_\Gamma(tid)$ or a pair fid, x such that $F(fid, x) \neq \widehat{F}(fid, x)$. Let $(G_\Delta, P, P^{-1}) = \phi(G, F, F^{-1})$ and $(\widehat{G}_\Delta, \widehat{P}, \widehat{P}^{-1}) = \phi(\widehat{G}, \widehat{F}, \widehat{F}^{-1})$. If there exists a tid such $G_\Gamma(tid) \neq \widehat{G}_\Gamma(tid)$, then $G_\Delta(tid) \neq \widehat{G}_\Delta(tid)$. Next, if there exist a pair fid, x such that $F(fid, x) \neq \widehat{F}(fid, x)$, then $P(fid, x) \neq \widehat{P}(fid, x)$. Thus $\phi(G, F, F^{-1}) \neq \phi(\widehat{G}, \widehat{F}, \widehat{F}^{-1})$ and the mapping is injective.

It remains to show that the mapping is onto. Let $O_\Delta = (G_\Delta, P, P^{-1})$ be a Δ-oracle. We show there exists a Γ-oracle O_Γ that fulfills $\phi(O_\Gamma) = O_\Delta$. Let O_Γ be the following oracle:

- For every $tid \in \{0,1\}^n$, it holds that $G_\Gamma(tid) = G_\Delta(tid)$.
- For every $fid \in \{0,1\}^{2n}$ and $x \in \{0,1\}^n$, it holds that $F(fid, x) = P(fid, x)$.
- For every $tid \in \{0,1\}^n$ and $y \in \{0,1\}^n$, it holds that $F^{-1}(tid, y) = P^{-1}(G_\Delta(tid), y)$.

First, observe that O_Γ is indeed a Γ-oracle:

- Since G_Δ is injective, it holds that G_Γ is injective.
- For every $fid \in Range(G_\Gamma)$, since $fid \in Range(G_\Delta)$ and $P(fid, \cdot)$ is a permutation over $\{0,1\}^n$, $F(fid, \cdot)$ is a permutation over $\{0,1\}^n$.
- For every $fid \notin Range(G_\Gamma)$, it holds that $fid \notin Range(G_\Delta)$ and for every $x \in \{0,1\}^n$, $P(fid, x) = \perp$ and thus $F(fid, x) = \perp$ as desired.
- For every fid, tid such that $fid = G_\Gamma(tid)$ and for every $x \in \{0,1\}^n$, $F^{-1}(tid, F(fid, x)) = P^{-1}(G_\Delta(tid), P(fid, x)) = P^{-1}(fid, P(fid, x)) = x$ as desired.

Verifying that $\phi(O_\Gamma) = O_\Delta$ is easy. □

The above claim immediately implies the following:

Corollary 4.2. *The random variables $\mathcal{U}_\Delta$ and $\phi(\mathcal{U}_\Gamma)$ are identically distributed.*

4.2. *Enhanced TDP Relative to* Γ*-Oracles*

In this section, we prove that enhanced trapdoor permutations exist relative to Γ-oracles. With every Γ-oracle $O_\Gamma = (G_\Gamma, F, F^{-1})$, we associate the following tuple of algorithms (I, S', F, B)[6]:

- On input 1^n, I selects a random $tid \in \{0,1\}^n$ and sets $(\alpha, \tau) = (G_\Gamma(tid), tid)$.
- On input α and a random $r \in \{0,1\}^n$, S' returns r (which is a uniformly distributed element in the domain of f_α).
- For a given α and x, algorithm F returns $f_\alpha(x) = F(\alpha, x)$.
- For a given τ and $y \in \{0,1\}^n$, algorithm B returns $F^{-1}(\tau, y)$. Note that if (α, τ) is in the range of $I(1^n)$, then $B(\tau, y) = f_\alpha^{-1}(y)$.

We prove that for almost all Γ-oracles, the tuple of algorithms defined above is an enhanced trapdoor permutation. We first show that f_α is hard to invert.

Claim 4.3 (Adapted from [27]). *Let M be an oracle PPT machine. Then, there exists a negligible function* $\mathrm{neg}(\cdot)$ *such that for any n and any input y of length n, and for every α in the range of $I_1(1^n)$:*

$$\Pr\left[f_\alpha\left(M^{\mathcal{U}_\Gamma}(\alpha, y)\right) = y\right] < \mathrm{neg}(n),$$

where the probability is taken over random Γ-oracles and over the random tape of M.

Proof. Machine M gets as input α and y and tries to invert f_α on y. Fix any random tape for M. First, we note that if M never queries G_Γ on a τ such that $G_\Gamma(\tau) = \alpha$, then it can guess the corresponding τ with probability at most $\frac{1}{2^n}$ (since τ is distributed uniformly on $\{0,1\}^n$). Now, note that every time M makes a query to $G_\Gamma(\tau)$, the probability that $G_\Gamma(\tau) = \alpha$ is equal to $\frac{1}{2^n}$. M makes only a polynomial number of queries to G_Γ, and therefore M can find the corresponding τ for α with only a negligible probability.

Now, we argue that if M does not possess the corresponding τ for α, then it can invert f_α on y with only a negligible probability. First, note that if M never queries $F(\alpha, \cdot)$ on an x such that $F(\alpha, x) = y$, then the probability that it outputs such an x is bounded by $\frac{1}{2^n}$ (since x is distributed uniformly on $\{0,1\}^n$). Every time M makes a query $F(\alpha, x)$, the probability that $F(\alpha, x) = y$ is equal to $\frac{1}{2^n}$. M makes only a polynomial number of queries to its oracle, and therefore the probability that $f_\alpha(M(\alpha, y)) = y$ is a negligible function of n. □

The following lemma is proven in [27, Sect. 4.2] using Markov's inequality and the Borel–Cantelli lemma.

Lemma 4.4 ([27]). *Let R be a set of oracles. If for every PPT oracle machine M there exists a negligible function* $\mathrm{neg}(\cdot)$ *such that for every n and every input corresponding*

[6] We use the definition of enhanced TDP that appears in [23], where I is a function that generates a random permutation index with a corresponding trapdoor, S is a function that given a permutation index samples a random element in its domain, F is a function that given a permutation index computes it on a given element in its domain, and B is the function that given a permutation trapdoor, returns the preimage of a given element in its domain.

to n, M succeeds in a given task with probability less than $\mathrm{neg}(n)$ (*when the probability is taken over random oracles in R and the random tape of M*), *then for measure* 1 *of oracles in R, for every PPT machine M there exists a negligible function* $\mathrm{neg}(\cdot)$ *such that for all sufficiently large n's, M succeeds in the given task with probability no more than* $\mathrm{neg}(n)$.

Combining Claim 4.3 and Lemma 4.4, we obtain the following theorem:

Theorem 4.5. *With probability* 1 *over random Γ-oracles, the algorithms (I, S', F, B) associated with the oracle form an enhanced trapdoor permutation. In particular, for every PPT machine M, every positive polynomial $p(\cdot)$ and for all sufficiently large n's,*

$$\Pr\left[f_{I_1(1^n)}\left(M\left(I_1\left(1^n\right), \mathcal{U}_n\right)\right) = \mathcal{U}_n\right] < \frac{1}{p(n)}.$$

We remark also that statically secure semi-honest oblivious transfer can be constructed from any enhanced trapdoor permutation [15] and thus exists relative to Γ-oracles.

4.3. *Static* OT_1^2 *Relative to* Δ*-Oracles from Adaptive* OT_1^2

In this section, we prove that if there exists an adaptively secure OT_1^2 relative to random Γ-oracles, then there exists a statically secure OT_1^2 relative to random Δ-oracles. We actually prove a more general theorem that if there exists a protocol for securely computing any functionality f in the presence of adaptive adversaries relative to a random Γ-oracle, then there exists a protocol for securely computing f in the presence of static adversaries relative to a random Δ-oracle. We restrict our proof to two-party protocols only, but stress that the claim can be proved similarly for multiparty protocols as well.

Let $\Pi_1 = \langle Alice_1, Bob_1 \rangle$ be a protocol for securely computing a functionality f in the presence of *adaptive* adversaries relative to a Γ-oracle. We use Π_1 to construct a new protocol $\Pi_2 = \langle Alice_2, Bob_2 \rangle$ for securely computing f in the presence of *static* adversaries relative to a Δ-oracle, where Π_2 is essentially the same as Π_1 with some minor changes because the parties in Π_2 have oracle access to a Δ-oracle (rather to a Γ-oracle in Π_1).

Recall that the parties $Alice_2$ and Bob_2 have access to a Δ-oracle, while in the original protocol, $Alice_1$ and Bob_1 have access to a Γ-oracle. There is a fundamental difference between these two cases because a Γ-oracle is inherently *asymmetric* (it is possible to send a party *fid* while keeping *tid* secret, thereby enabling them to compute the permutation but not invert it), while a Δ-oracle is inherently *symmetric* (the same *fid* is used to compute and invert the permutation). The idea behind our proof is to eliminate the asymmetric nature of the Γ-oracle by using the fact that in the adaptive setting (e.g., in the post-execution corruption phase), the distinguisher can ultimately corrupt all parties (for example, if Bob is corrupted in an execution of the statically secure protocol, then in the adaptive setting, we corrupt Bob at the beginning of the protocol and corrupt Alice in the post-execution phase). If it does so, it obtains the entire view of all parties and in particular the view of any party who samples a permutation using G_Γ. The critical observation is that since the range of G_Γ is sparse, the probability of a party finding an

fid in the range of G_Γ without explicitly querying it is negligible.[7] However, if it does make such a query, then its view contains *both fid and tid* and this will be obtained by the distinguisher upon corrupting the parties. Thus, the distinguisher is able to compute and invert the permutation, just like in the case of a Δ-oracle. The fact that the adaptive simulator must simulate well even when the distinguisher works in this way (learning all *fid*, *tid* pairs) is the basis for constructing a simulator for the static case when using a Δ-oracle.

We begin by defining $\Pi_2 = \langle Alice_2, Bob_2 \rangle$ which is constructed from Π_1 by replacing the Γ-oracle with a Δ-oracle:

Protocol Π_2. On input x_A, $Alice_2$ invokes $Alice_1$ on x_A. On input x_B, Bob_2 invokes Bob_1 on x_B. The execution is described below for a party $\mathcal{P}_2$ emulating $\mathcal{P}_1$, and is the same for both $Alice_2$ and Bob_2. In each round,

- When $\mathcal{P}_2$ gets the message sent by the other party in the previous round, it hands it to $\mathcal{P}_1$.
- If $\mathcal{P}_1$ makes a query *tid* to the oracle G_Γ, $\mathcal{P}_2$ queries $G_\Delta(tid)$ and gets an output *fid* and hands *fid* to $\mathcal{P}_1$.
- If $\mathcal{P}_1$ makes a query (fid, x) to F, $\mathcal{P}_2$ queries $P(fid, x)$, receives an output y and hands y to $\mathcal{P}_1$ (note that y may equal $\perp$).
- If $\mathcal{P}_1$ makes a query (tid, y) to F^{-1}, $\mathcal{P}_2$ first queries its oracle G_Δ on *tid* and receives an output *fid*. $\mathcal{P}_2$ then queries $P^{-1}(fid, y)$, obtains an output x and hands x to $\mathcal{P}_1$.
- If $\mathcal{P}_1$ writes a string m on its outgoing communication tape, $\mathcal{P}_2$ sends m to the other party.
- At the end of the simulation, $\mathcal{P}_2$ outputs the output of $\mathcal{P}_1$.

We now prove that Π_2 securely computes the functionality f in the presence of semi-honest static adversaries.

Theorem 4.6. *If Π_1 securely computes a functionality f in the presence of adaptive adversaries relative to a random Γ-oracle O_Γ, then Π_2 securely computes f in the presence of static semi-honest adversaries relative to the Δ-oracle $\phi(O_\Gamma)$.*

Proof. The intuition has already been described above and we therefore proceed directly to the proof. Let O_Γ be an oracle that is distributed according to $\mathcal{U}_\Gamma$. We show that if Π_1 is a secure adaptive protocol for computing f relative to O_Γ, then Π_2 is a secure static semi-honest protocol for computing f relative to $O_\Delta = \phi(O_\Gamma)$. It is easy to see that Π_2 computes f relative to O_Δ because an execution of Π_2 is, in fact, an execution of Π_1 with a simulated Γ-oracle which is exactly $\phi^{-1}(O_\Delta) = O_\Gamma$.

Next we turn to the security of Π_2. We show that $\Pi_2 = \langle Alice_2, Bob_2 \rangle$ securely computes the functionality f in the presence of static semi-honest adversaries relative to a $O_\Delta = \phi(O_\Gamma)$. We use the ideal-process simulator $\mathcal{SIM}$ of Π_1 for the adaptive setting

[7] Note that we know how to construct adaptively secure OT from enhanced trapdoor permutations if the functions are dense. The construction follows from [14] and [9].

to construct two probabilistic polynomial-time simulators $\mathcal{S}_{Alice_2}$ and $\mathcal{S}_{Bob_2}$ for Π_2 in the static setting, such that

$$\left\{\mathcal{S}_{Alice_2}^{O_\Delta}(1^n, x_A, y_A)\right\}_{x_A,x_B\in\{0,1\}^*} \overset{c^O}{\equiv} \left\{\mathrm{VIEW}_{Alice_2}^{\Pi_2,O_\Delta}(1^n, x_A, x_B)\right\}_{x_A,x_B\in\{0,1\}^*},$$

$$\left\{\mathcal{S}_{Bob_2}^{O_\Delta}(1^n, x_B, y_B)\right\}_{x_A,x_B\in\{0,1\}^*} \overset{c^O}{\equiv} \left\{\mathrm{VIEW}_{Bob_2}^{\Pi_2,O_\Delta}(1^n, x_A, x_B)\right\}_{x_A,x_B\in\{0,1\}^*}.$$

We use the ideal-process simulator $\mathcal{SIM}$ of Π_1 for the adaptive setting (see Sect. 2.2) to construct two probabilistic polynomial-time simulators $\mathcal{S}_{Alice_2}$ and $\mathcal{S}_{Bob_2}$. Since the constructions of $\mathcal{S}_{Alice_2}$ and $\mathcal{S}_{Bob_2}$ and the proofs of indistinguishability are very similar, we present only a construction and a proof for $\mathcal{S}_{Bob_2}$.

Let $\mathcal{A}$ and $\mathcal{Z}$ be the following adversary strategy and environment: $\mathcal{Z}$ starts with an input $z \in \{0,1\}$. At the onset of the run of Π_1, $\mathcal{A}$ corrupts Bob_1 and at the end of the computation outputs the entire view of Bob_1. In the postexecution phase, if $z = 0$, no corruptions are made and if $z = 1$, $\mathcal{Z}$ creates a "corrupt $Alice_1$" message, hands it to $\mathcal{A}$ who corrupts Alice. Eventually $\mathcal{Z}$ outputs the entire view of the corrupted parties (that is, if $z = 0$, the view of Bob alone and if $z = 1$, the view of both parties). No auxiliary information is sent by $\mathcal{Z}$ to $\mathcal{A}$. Let $\mathcal{SIM}$ be the ideal-process adversary guaranteed to exist for $\mathcal{A}$ and $\mathcal{Z}$ by the adaptive security of Π_1. We now use $\mathcal{A}$, $\mathcal{Z}$ and $\mathcal{SIM}$ to define $\mathcal{S}_{Bob_2}$ (the static simulator for the case that Bob is corrupted). $\mathcal{S}_{Bob_2}$ receives the input x_B and output y_B of Bob as defined by the functionality f and emulates the run of $\mathcal{SIM}$ in the adaptive ideal model with environment $\mathcal{Z}$ with input $z = 0$. Note that $\mathcal{SIM}$ must corrupt only Bob, because in the real world only Bob is corrupted when $z = 0$. We also can assume w.l.o.g. that $\mathcal{SIM}$ corrupts Bob in the first corruption phase.

$\mathcal{S}_{Bob_2}$ receives input (x_B, y_B) and works as follows, simulating a Γ-oracle for $\mathcal{SIM}$ using its Δ-oracle:

- If $\mathcal{SIM}$ makes a query r to the oracle G_Γ, $\mathcal{S}_{Bob_2}$ queries its oracle $G_\Delta(r)$, receives an output *fid* and hands it to $\mathcal{SIM}$.
- If $\mathcal{SIM}$ makes a query (fid, x) to F, $\mathcal{S}_{Bob_2}$ queries it oracle $P(fid, x)$, gets an output y and hands it to $\mathcal{SIM}$.
- If $\mathcal{SIM}$ makes a query (tid, y) to F^{-1}, $\mathcal{S}_{Bob_2}$ first queries its oracle G_Δ on *tid*, gets an output *fid*. $\mathcal{S}_{Bob_2}$ then queries $P^{-1}(fid, y)$, gets an output x and hands x to $\mathcal{SIM}$.
- When $\mathcal{SIM}$ decides to corrupt Bob_1, $\mathcal{S}_{Bob_2}$ plays the role of $\mathcal{Z}$ by sending x_B to $\mathcal{SIM}$.
- In the computation phase, $\mathcal{S}_{Bob_2}$ plays the role of the trusted party and sends y_B to $\mathcal{SIM}$ (recall that $\mathcal{S}_{Bob_2}$ gets y_B as input).
- At the end of the simulation, $\mathcal{S}_{Bob_2}$ outputs the output of $\mathcal{SIM}$.

Informally speaking, we show that a distinguisher D_2 for Π_2 and $\mathcal{S}_{Bob_2}$ (relative to O_Δ) implies the existence of a distinguisher D_1 for Π_1 and $\mathcal{SIM}$ (relative to O_Γ). The idea is to have D_1 simulate the run of D_2 on the view of Bob. However, D_2 has oracle access to a Δ-oracle O_Δ, while D_1 has oracle access to a Γ-oracle O_Γ. This might be problematic, for example, if D_2 wishes to compute $P^{-1}(fid, y)$ but D_1 doesn't know the corresponding *tid* (recall that D_1 can only invert y in the Γ-oracle world if it holds the

trapdoor *tid* whereas D_2 can invert y given *fid* only). Despite the above, we use the fact that the range of G_Γ is a negligible fraction of $\{0,1\}^{2n}$, and therefore any *fid* used in the protocol (except with negligible probability) must have been generated via a query to G_Γ, as described in the intuition above. More specifically, we show that if there exists a distinguisher D_2 that distinguishes the output of $\mathcal{S}_{Bob_2}$ from the output of a corrupted Bob_2 in a real execution of Π_2, then there exists a distinguisher D_1 that distinguishes the result of an ideal execution with $\mathcal{SIM}$ from a real execution of Π_1 with adversary $\mathcal{A}$ and environment $\mathcal{Z}$ *with input* $z=1$, meaning that Alice is also corrupted at the end. (Note that we set $z=0$ in order to define $\mathcal{S}_{Bob_2}$, but now set $z=1$ to construct the distinguisher. Since $\mathcal{SIM}$ has to work for all inputs z to $\mathcal{Z}$, this suffices.) Since both Alice and Bob are corrupted in this execution, D_1 obtains all of the (*fid*, *tid*) pairs generated by queries to G_Γ and so it can invert always, enabling it to run D_2 and use its Γ-oracle to answer all of D_2's Δ queries.

Formally, assume that there exist a PPT machine D_2 and a positive polynomial $p(\cdot)$ such that for infinitely many n's, there exist $x_A, x_B \in \{0,1\}^*$ such that

$$\begin{aligned}&\left|\Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathrm{VIEW}_{Bob_2}^{\Pi_2, O_\Delta}\left(1^n, x_A, x_B\right)\right)=1\right]\right.\\&\left.-\Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathcal{S}_{Bob_2}^{O_\Delta}\left(1^n, x_B, y_B\right)\right)=1\right]\right| \geq \frac{1}{p(n)}. \qquad (1)\end{aligned}$$

We use D_2 to construct a PPT machine D_1 and a positive polynomial $q(\cdot)$ such that for infinitely many n's, there exist $x_A, x_B \in \{0,1\}^*, z \in \{0,1\}$ such that

$$\begin{aligned}&\left|\Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, z, \mathrm{REAL}_{\Pi_1,\mathcal{A},\mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, z)\right)=1\right]\right.\\&\left.-\Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, z, \mathrm{IDEAL}_{\mathrm{OT}_1^2,\mathcal{SIM},\mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, z)\right)=1\right]\right| \geq \frac{1}{q(n)}.\end{aligned}$$

The distinguisher D_1 receives as input x_A, x_B, z and the output of $\mathcal{Z}$. If $z=0$, D_1 returns 0 and halts. Otherwise, the output of $\mathcal{Z}$ consists of both parties' views, including *Alice*'s input x_A and random tape r_A and *Bob*'s input x_B and random tape r_B. D_1 begins by initializing a table T that will hold all pairs (*fid*, *tid*) generated by queries to the oracle. D_1 invokes $\langle Alice_1^{O_\Gamma}, Bob_1^{O_\Gamma}\rangle$ on the appropriate input and random tapes (recall that they are a part of D_1's input because $\mathcal{A}$ outputs the views of both parties when $z=1$) and for every access of one of the parties to G_Γ, namely a query $G_\Gamma(tid)=fid$, D_1 records the entry (*fid*, *tid*) in T. When it finishes the execution of $\langle Alice_1^{O_\Gamma}, Bob_1^{O_\Gamma}\rangle$, D_1 starts simulating D_2 on the view of Bob and proceeds as follows:

- If D_2 makes a query $G_\Delta(tid)$, D_1 makes a query $G_\Gamma(tid)$, gets an *fid*, records the entry (*fid*, *tid*) in T and hands *fid* to D_2.
- If D_2 tries to compute $P(fid, x)$, D_1 queries its oracle $F(fid, x)$ and returns its answer.
- If D_2 tries to compute $P^{-1}(fid, y)$, D_1 checks whether an entry (*fid*, *tid*) exists in T. If not, it returns $\perp$. If yes, it queries $F^{-1}(tid, y)$ and returns its answer.

Note that if $z=1$, a run of D_1 on input x_A, x_B, z and the views of both parties with access to an oracle O_Γ is a simulation of the run of D_2 on input x_B and the view of *Bob*

with a simulated Δ-oracle obtained from O_Γ. It's easy to see that the only differences between the simulated Δ-oracle obtained from O_Γ by D_1 and $\phi(O_\Gamma)$ are queries to P^{-1} on an *fid* in the range of G_Δ that was not created via a query to G_Δ: For such queries, the simulated Δ-oracle replies with $\perp$ since a pair (fid, tid) doesn't exist in T, while $\phi(O_\Gamma)$'s answer is different from $\perp$. We define the event FIND to be true if and only if D_2 makes a query to its P^{-1} oracle involving an *fid* that was not created via a query to G_Δ. Note that finding an *fid* in the range of G_Δ without making a query to G_Δ can happen with only a negligible probability (recall that the range of $G_\Delta$is a negligible fraction of $\{0,1\}^{2n}$), and therefore

$$\Pr[\mathsf{FIND}] < \frac{1}{p^2(n)}.$$

We now examine the behavior of D_1 in case $\mathsf{FIND} = \mathit{false}$ and $z = 1$ and show that there exists a polynomial $q(\cdot)$ such that for infinitely many n's, there exist $x_A, x_B \in \{0,1\}^*$ such that

$$\begin{aligned}&\Big|\Pr\Big[D_1^{O_\Gamma}\big(1^n, x_A, x_B, 1, \mathrm{REAL}^{O_\Gamma}_{\Pi_1,\mathcal{A},\mathcal{Z}}(n, x_A, x_B, 1)\big) = 1\Big]\\ &\quad - \Pr\Big[D_1^{O_\Gamma}\big(1^n, x_A, x_B, 1, \mathrm{IDEAL}^{O_\Gamma}_{\mathrm{OT}^2_1,\mathcal{SIM},\mathcal{Z}}(n, x_A, x_B, 1)\big) = 1\Big]\Big| \geq \frac{1}{q(n)}.\end{aligned}$$

Recall that for infinitely many n's, there exist $x_A, x_B \in \{0,1\}^*$ for which (1) holds. Fix such an n and the corresponding $x_A, x_B \in \{0,1\}^*$.

First, assume that the input of D_1 is the random variable $\mathrm{REAL}^{O_\Gamma}_{\Pi_1,\mathcal{A},\mathcal{Z}}(n, x_A, x_B, 1)$:

$$\begin{aligned}&\Pr\Big[D_1^{O_\Gamma}\big(1^n, x_A, x_B, \mathrm{REAL}^{O_\Gamma}_{\Pi_1,\mathcal{A},\mathcal{Z}}(n, x_A, x_B, 1)\big) = 1\Big]\\ &\quad = \Pr\Big[D_1^{O_\Gamma}\big(1^n, x_A, x_B, \mathrm{REAL}^{O_\Gamma}_{\Pi_1,\mathcal{A},\mathcal{Z}}(n, x_A, x_B, 1)\big) = 1 | \neg\mathsf{FIND}\Big] \cdot \Pr[\neg\mathsf{FIND}]\\ &\qquad + \Pr\Big[D_1^{O_\Gamma}\big(1^n, x_A, x_B, \mathrm{REAL}^{O_\Gamma}_{\Pi_1,\mathcal{A},\mathcal{Z}}(n, x_A, x_B, 1)\big) = 1 | \mathsf{FIND}\Big] \cdot \Pr[\mathsf{FIND}]\\ &\quad \geq \Pr\Big[D_1^{O_\Gamma}\big(1^n, x_A, x_B, \mathrm{REAL}^{O_\Gamma}_{\Pi_1,\mathcal{A},\mathcal{Z}}(n, x_A, x_B, 1)\big) = 1 | \neg\mathsf{FIND}\Big] \cdot \Pr[\neg\mathsf{FIND}].\end{aligned}$$

In this case, the view of Bob_1 that is contained in the input of D_1 is also a view of Bob_2 in a real-world run of $\Pi_2(1^n, x_A, x_B)$ with oracle access to $O_\Delta = \phi(O_\Gamma)$, since a run of Π_2 with oracle access to a Δ-oracle O_Δ is, in fact, a simulation of a run of Π_1 with oracle access to a Γ-oracle $\phi^{-1}(O_\Delta)$. Recall that when $z = 1$, D_1 with oracle access to a Γ-oracle O_Δ invokes a run of D_2 with a simulated Δ-oracle. If $\mathsf{FIND} = \mathit{false}$, D_1 returns 1 on $\mathrm{REAL}^{O_\Gamma}_{\Pi_1,\mathcal{A},\mathcal{Z}}(n, x_A, x_B, 1)$ if and only if D_2 returns 1 on $\mathrm{VIEW}^{\Pi_2, O_\Delta}_{Bob_2}(1^n, x_A, x_B)$, and therefore

$$\begin{aligned}&\Pr\Big[D_1^{O_\Gamma}\big(1^n, x_A, x_B, \mathrm{REAL}^{O_\Gamma}_{\Pi_1,\mathcal{A},\mathcal{Z}}(n, x_A, x_B, 1)\big) = 1 | \neg\mathsf{FIND}\Big]\\ &\quad = \Pr\Big[D_2^{O_\Delta}\big(1^n, x_A, x_B, \mathrm{VIEW}^{\Pi_2, O_\Delta}_{Bob_2}\big(1^n, x_A, x_B\big)\big) = 1 | \neg\mathsf{FIND}\Big].\end{aligned}$$

Now,

$$\begin{aligned}
&\Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathrm{VIEW}_{Bob_2}^{\Pi_2, O_\Delta}\left(1^n, x_A, x_B\right)\right) = 1 | \neg \mathsf{FIND}\right] \cdot \Pr[\neg \mathsf{FIND}] \\
&\quad = \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathrm{VIEW}_{Bob_2}^{\Pi_2, O_\Delta}\left(1^n, x_A, x_B\right)\right) = 1\right] \\
&\qquad - \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathrm{VIEW}_{Bob_2}^{\Pi_2, O_\Delta}\left(1^n, x_A, x_B\right)\right) = 1 | \mathsf{FIND}\right] \cdot \Pr[\mathsf{FIND}] \\
&\quad \geq \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathrm{VIEW}_{Bob_2}^{\Pi_2, O_\Delta}\left(1^n, x_A, x_B\right)\right) = 1\right] - \Pr[\mathsf{FIND}].
\end{aligned}$$

We obtain that

$$\begin{aligned}
&\Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \mathrm{REAL}_{\Pi_1, \mathcal{A}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, 1)\right) = 1\right] \\
&\quad \geq \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathrm{VIEW}_{Bob_2}^{\Pi_2, O_\Delta}\left(1^n, x_A, x_B\right)\right) = 1\right] - \Pr[\mathsf{FIND}].
\end{aligned}$$

Now, assume that the input of D_1 is the value of the random variable $\mathrm{IDEAL}_{\mathrm{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, z)$:

$$\begin{aligned}
&\Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \mathrm{IDEAL}_{\mathrm{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, 1)\right) = 1\right] \\
&\quad = \Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \mathrm{IDEAL}_{\mathrm{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, 1)\right) = 1 | \neg \mathsf{FIND}\right] \cdot \Pr[\neg \mathsf{FIND}] \\
&\qquad + \Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \mathrm{IDEAL}_{\mathrm{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, 1)\right) = 1 | \mathsf{FIND}\right] \cdot \Pr[\mathsf{FIND}] \\
&\quad \leq \Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \mathrm{IDEAL}_{\mathrm{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, 1)\right) = 1 | \neg \mathsf{FIND}\right] \cdot \Pr[\neg \mathsf{FIND}] \\
&\qquad + \Pr[\mathsf{FIND}].
\end{aligned}$$

In this case, the view of Bob_1 that is contained in the input of D_1 is also a view of Bob_2 created by $\mathcal{S}_{Bob_2}$ with oracle access to $O_\Delta = \phi(O_\Gamma)$. This claim is true because $\mathcal{S}_{Bob_2}$ emulates a run of $\mathcal{SIM}$ with $\mathcal{Z}$ and $z = 0$ and trusted party $\mathcal{T}$ with a simulated Γ-oracle that is equal to $\phi^{-1}(O_\Delta) = O_\Gamma$ and outputs the view of Bob created by $\mathcal{SIM}$. Since the view of Bob depends only on the execution phase, the view of Bob that is contained in the output of $\mathrm{IDEAL}_{\mathrm{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, z)$ is equal to the output of $\mathcal{S}_{Bob_2}^{O_\Delta}(1^n, x_B, s_{x_B})$ regardless of the value of z. Therefore, if $\mathsf{FIND} = \mathit{false}$, D_1 outputs 1 on $\mathrm{IDEAL}_{\mathrm{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, 1)$ if and only if D_2 returns 1 on $\mathcal{S}_{Bob_2}^{O_\Delta}(1^n, x_B, s_{x_B})$, and therefore

$$\begin{aligned}
&\Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \mathrm{IDEAL}_{\mathrm{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, 1)\right) = 1 | \neg \mathsf{FIND}\right] \\
&\quad = \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathcal{S}_{Bob_2}^{O_\Delta}\left(1^n, x_B, s_{x_B}\right)\right) = 1 | \neg \mathsf{FIND}\right].
\end{aligned}$$

Now,

$$\begin{aligned}
&\Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathcal{S}_{Bob_2}^{O_\Delta}\left(1^n, x_B, s_{x_B}\right)\right) = 1 | \neg \mathsf{FIND}\right] \cdot \Pr[\neg \mathsf{FIND}] \\
&\quad = \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathcal{S}_{Bob_2}^{O_\Delta}\left(1^n, x_B, s_{x_B}\right)\right) = 1\right] \\
&\qquad - \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathcal{S}_{Bob_2}^{O_\Delta}\left(1^n, x_B, s_{x_B}\right)\right) = 1 | \mathsf{FIND}\right] \cdot \Pr[\mathsf{FIND}] \\
&\quad \leq \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathcal{S}_{Bob_2}^{O_\Delta}\left(1^n, x_B, s_{x_B}\right)\right) = 1\right].
\end{aligned}$$

We obtain

$$\begin{aligned}
&\Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \text{IDEAL}_{\text{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, 1)\right) = 1\right] \\
&\qquad \leq \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathcal{S}_{Bob_2}^{O_\Delta}\left(1^n, x_B, s_{x_B}\right)\right) = 1\right] + \Pr[\mathsf{FIND}].
\end{aligned}$$

We conclude that

$$\begin{aligned}
&\left|\Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \text{REAL}_{\Pi_1, \mathcal{A}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, z)\right) = 1\right]\right. \\
&\qquad \left. - \Pr\left[D_1^{O_\Gamma}\left(1^n, x_A, x_B, \text{IDEAL}_{\text{OT}_1^2, \mathcal{SIM}, \mathcal{Z}}^{O_\Gamma}(n, x_A, x_B, z)\right) = 1\right]\right| \\
&\quad \geq \left|\Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \text{VIEW}_{Bob_2}^{\Pi_2, O_\Delta}\left(1^n, x_A, x_B\right)\right) = 1\right] - \Pr[\mathsf{FIND}]\right. \\
&\qquad \left. - \Pr\left[D_2^{O_\Delta}\left(1^n, x_A, x_B, \mathcal{S}_{Bob_2}^{O_\Delta}\left(1^n, x_B, s_{x_B}\right)\right) = 1\right] - \Pr[\mathsf{FIND}]\right| \\
&\quad > \frac{1}{p(n)} - \frac{2}{p^2(n)} \geq \frac{1}{q(n)}
\end{aligned}$$

for some positive polynomial $q(\cdot)$. We conclude that $\Pi_2 = \langle Alice_2, Bob_2 \rangle$ is statically secure relative to O_Δ, and this completes the proof of Theorem 4.6. □

Remark 4.7. Theorem 4.6 is true only for *random* Γ-oracles. Specifically, if O_Γ is not a random Γ-oracle, then the claim that finding an *fid* in the range of G_Γ without making a query to it can happen only with negligible probability does not necessarily hold, and therefore the theorem is not necessarily true for an arbitrary Γ-oracle.

The following corollary is obtained from Theorem 4.6 by taking oblivious transfer as a special case:

Corollary 4.8. *If there exists a protocol* Π_1 *that securely computes the* OT_1^2 *functionality in the presence of adaptive semi-honest adversaries relative to a random* Γ*-oracle* O_Γ*, then the protocol* Π_2 *defined above securely computes the* OT_1^2 *functionality in the presence of static semi-honest adversaries relative to* $\phi(O_\Gamma)$.

4.4. *No Static* OT_1^2 *Relative to* Δ*-Oracles*

For the next step of our proof, we show that static OT_1^2 does not exist relative to most Δ-oracles. In order to do this, we show that key agreement does not exist relative to most

Δ-oracles, and then derive the result from the fact that secure OT_1^2 implies key agreement. In order to show that key agreement does not exist relative to most Δ-oracles, we show that a Δ-oracle can be replaced with a "plain random oracle", with at most a negligible difference. Thus, the results of [27] for key agreement relative to a plain random oracle hold also relative to a Δ-oracle. We begin by formally defining a random oracle type, denoted ρ, and show its relationship to Δ-oracles.

ρ-Oracles We define a ρ-oracle to be an oracle with the following functions:

- G_ρ is an injective function from $\{0,1\}^n$ to $\{0,1\}^{2n}$.
- F_P is a function that on a triple $(I,k,x) \in \{0,\ldots,5\} \times \{0,1\}^{2n} \times \{0,1\}^{\frac{n}{2}}$ such that $k \in Range(G_\rho)$ returns a string $y \in \{0,1\}^{\frac{n}{2}}$. If $k \notin Range(G_\rho)$, F_P returns $\perp$. Note that for a given I and $k \in Range(G_\rho)$, $F_P(I,k,\cdot)$ is a function from $\{0,1\}^{\frac{n}{2}}$ to $\{0,1\}^{\frac{n}{2}}$.

Note that the output of G_ρ is an *fid*—or symmetric key k—of length $2n$ which defines 6 random functions $F_P(0,k,\cdot),\ldots,F_P(5,k,\cdot)$. The reason for this is that to show that key-agreement does not exist relative to Δ-oracles, we use the result of [11] who show that using 6 rounds of the Luby–Rackoff construction, the random oracle model and the ideal cipher model are equivalent. Hence $F_P(0,k,\cdot),\ldots,F_P(5,k,\cdot)$ are used to simulate the P permutation of a Δ-oracle, using 6 different functions for Luby–Rackoff.

We denote by $\mathcal{U}_\rho$ the uniform distribution on ρ-oracles. Namely, we say that a ρ-oracle $O_\rho = (G_\rho, F_P)$ is distributed according to $\mathcal{U}_\rho$ if G_ρ is a uniformly distributed injective function from $\{0,1\}^n$ to $\{0,1\}^{2n}$ and F_P is a uniformly distributed function from $\{0,\ldots,5\} \times Range(G_\rho) \times \{0,1\}^{\frac{n}{2}}$ to $\{0,1\}^{\frac{n}{2}}$ (and for k not in the range of G_ρ, it returns $\perp$). We sometimes use the phrase "O_ρ is a random ρ-oracle" as an abbreviation for "O_ρ is distributed according to $\mathcal{U}_\rho$".

We now prove the following theorem:

Theorem 4.9. *If $\mathcal{P} = \mathcal{NP}$, then relative to measure 1 of Δ-oracles, there does not exist any statically secure protocol for computing the OT_1^2 functionality.*

In order to prove Theorem 4.9, we recall the original black-box separation of key agreement from a random oracle, as proven in [27]. In [27], the black-box separation was proved by showing that if $\mathcal{P} = \mathcal{NP}$, then relative to a random oracle, for every key-agreement there exists a PPT machine *Eve* that can find a polynomial size list that contains all intersection queries with a good probability (where an "intersection query" is a query made by both *Alice* and *Bob*). Now, we can assume that at the end of the protocol, both *Alice* and *Bob* query their oracle on the key they are about to output. Therefore, if *Eve* can find all intersection queries, she can guess the key with a high probability. The same proof can be extended to oracles containing random G_ρ and F_P as defined above (see Footnote 9 in [19]).

Theorem 4.10 ([19,27]). *If $\mathcal{P} = \mathcal{NP}$, then given any key-agreement protocol relative to a random ρ-oracle, for every polynomial* poly$(\cdot)$, *there exists a polynomial time Eve such that Eve finds all intersection queries with probability* $1 - \frac{1}{\mathrm{poly}(n)}$.

We first show that a similar argument holds relative to Δ-oracles (which essentially represent symmetric encryption). That is, we show that every key agreement protocol relative to a random Δ-oracle can be broken with probability $1 - \frac{1}{\text{poly}(n)}$. To do this, we use the result of [11] that shows an equivalence between the random oracle model (represented by ρ-oracles in our work) and the ideal cipher model (represented by Δ-oracles in our work).

Theorem 4.11 ([11]). *There exists a Turing machine C such that $C^{\mathcal{U}_\rho}$ is equivalent to $\mathcal{U}_\Delta$. That is, for every polynomial q there exists a probabilistic polynomial-time simulator $\mathcal{S}$ and a negligible function μ, such that for every (possibly unbounded) machine D making at most $q(n)$ queries,*

$$\left|\Pr\left[D^{\mathcal{U}_\rho, C^{\mathcal{U}_\rho}}\left(1^n\right) = 1\right] - \Pr\left[D^{\mathcal{S}^{\mathcal{U}_\Delta}, \mathcal{U}_\Delta}\left(1^n\right) = 1\right]\right| < \mu(n).$$

We show that Theorem 4.11 implies the following:

Proposition 4.12. *If $\mathcal{P} = \mathcal{NP}$, then given any key-agreement protocol relative to a random Δ-oracle, for every polynomial* poly$(\cdot)$, *there exists a polynomial time Eve such that Eve finds all intersection queries with probability $1 - \frac{1}{2\text{poly}(n)}$.*

Proof Sketch: Let $\langle \mathcal{A}_1, \mathcal{B}_1 \rangle$ be a key-agreement protocol relative to random Δ-oracles. We use $\langle \mathcal{A}_1, \mathcal{B}_1 \rangle$ to construct a key-agreement protocol $\langle \mathcal{A}_2, \mathcal{B}_2 \rangle$ relative to random ρ-oracles. Recall that $\mathcal{A}_2$ and $\mathcal{B}_2$ have oracle access to a ρ-oracle while $\mathcal{A}_1$ and $\mathcal{B}_1$ have oracle access to a Δ-oracle. The idea is to use the ρ-oracle in order to simulate a Δ-oracle using the Turing machine C that is guaranteed to exist by Theorem 4.11. Now, assume $\mathcal{P} = \mathcal{NP}$. Let poly$(\cdot)$ be some polynomial and let Eve_2 be as in Theorem 4.10. We use Eve_2 to construct an adversary Eve_1 for $\langle \mathcal{A}_1, \mathcal{B}_1 \rangle$. Eve_1 simply invokes Eve_2 and simulates the ρ-oracle using the simulator $\mathcal{S}$ guaranteed to exist by Theorem 4.11. Note that if Eve_1 outputs a list of intersection queries with probability less than $1 - \frac{1}{2\text{poly}(n)}$, then it is possible to distinguish oracles $(\mathcal{U}_\rho, C^{\mathcal{U}_\rho})$ from $(\mathcal{S}^{\mathcal{U}_\Delta}, \mathcal{U}_\Delta)$ with non-negligible probability. Specifically, given a pair of oracles $(\mathcal{O}_1, \mathcal{O}_2)$ that are distributed according to $(\mathcal{U}_\rho, C^{\mathcal{U}_\rho})$ or $(\mathcal{S}^{\mathcal{U}_\Delta}, \mathcal{U}_\Delta)$, distinguisher D first invokes a run of $\langle \mathcal{A}_1^{\mathcal{O}_2}, \mathcal{B}_1^{\mathcal{O}_2} \rangle$ and then invokes $Eve_2^{\mathcal{O}_1}$ on the transcript. D outputs 1 if and only if Eve_2 outputs all intersection queries. Now, if $(\mathcal{O}_1, \mathcal{O}_2)$ are distributed according to $(\mathcal{U}_\rho, C^{\mathcal{U}_\rho})$ then Eve_2 outputs all intersection queries with probability at least $1 - \frac{1}{\text{poly}(n)}$, and if $(\mathcal{O}_1, \mathcal{O}_2)$ are distributed according to $(\mathcal{S}^{\mathcal{U}_\Delta}, U_\Delta)$ then Eve_2 outputs all intersection queries with probability less than $1 - \frac{1}{2\text{poly}(n)}$. Thus D distinguishes with non-negligible probability. □

Remark 4.13. Theorem 4.11 holds even when $\mathcal{P} = \mathcal{NP}$ since the running time of D is unbounded.

Now we can use the same methods as in [27] to show that relative to measure 1 of Δ-oracles, any key-agreement can be broken in polynomial time. As described in [19], it is possible to construct a secure key agreement from any static oblivious transfer protocol and it is easy to verify that this construction relativizes. Therefore, we conclude that

relative to measure 1 of Δ-oracles, there does not exist any statically secure protocol for computing the OT_1^2 functionality.

The following corollary can be proven using the same methods as in [27] (the only difference between it and what was proven in [27] is the type of oracle used and this is irrelevant for the proof because this corollary only shows that if any key-agreement protocol relative to a *random oracle* can be broken with probability as high as $1 - 1/\mathrm{poly}(n)$, then this is true for *measure 1* of oracles):

Corollary 4.14. *If $\mathcal{P} = \mathcal{NP}$, then for measure 1 of Δ-oracles, any key-agreement protocol can be broken in polynomial time.*

Recalling that the existence of a secure OT_1^2 relative to an oracle $\mathcal{O}$ implies the existence of a secure key agreement relative to $\mathcal{O}$, we obtain:

Corollary 4.15. *If $\mathcal{P} = \mathcal{NP}$, then for measure 1 of Δ-oracles, there does not exist any statically secure protocol for computing the OT_1^2 functionality.*

4.5. *Concluding the Proof*

Corollary 4.8 states that if there exists an adaptively secure protocol for OT_1^2 relative to a given Γ-oracle $\mathcal{O}$, then there exists a statically secure protocol for OT_1^2 relative to the oracle $\phi(\mathcal{O})$. Now, by Theorem 4.9, for measure 1 of Δ-oracles, there exists no statically secure OT_1^2. Using the fact that ϕ is a bijection (Claim 4.1), we conclude that for measure 1 of Γ-oracles, there exists no adaptively secure OT_1^2. That is, we have the following:

Theorem 4.16. *If $\mathcal{P} = \mathcal{NP}$, then for measure 1 of Γ-oracles, there does not exist any adaptively secure protocol for computing the OT_1^2 functionality.*

Similarly as in [27], we derive an oracle separation of enhanced trapdoor permutations from adaptively secure OT_1^2 (even for semi-honest adversaries):

Corollary 4.17. *There exists an oracle relative to which enhanced trapdoor permutations exist, but not adaptively secure OT_1^2.*

Proof. Let $\mathcal{O}$ be a $\mathcal{PSPACE}$-complete oracle combined with a random Γ-oracle. Enhanced trapdoor permutations exist relative to $\mathcal{O}$, whereas adaptively secure OT_1^2 does not, as we have shown. □

Acknowledgements

We thank Omer Reingold for helpful discussions.

References

[1] D. Beaver, Adaptive zero knowledge and computational equivocation, in *28th STOC* (1996), pp. 629–638

[2] D. Beaver, Plug and play encryption, in *Advances in Cryptology—Crypto '97* (1997), pp. 75–89
[3] D. Beaver, Adaptively secure oblivious transfer, in *ASIACRYPT'98*. LNCS, vol. 1514 (Springer, Berlin, 1998), pp. 300–314
[4] M. Bellare, S. Micali, R. Ostrovsky, Perfect zero-knowledge in constant rounds, in *22nd STOC* (1990), pp. 482–493
[5] M. Blum, How to prove a theorem so no one else can claim it, in *Proceedings of the International Congress of Mathematicians*, USA, pp. 1444–1451
[6] R. Canetti, Security and composition of multiparty cryptographic protocols. *J. Cryptol.* **13**(1), 143–202 (2000)
[7] R. Canetti, M. Fischlin, Universally composable commitments, in *CRYPTO 2001*. LNCS, vol. 2139 (Springer, Berlin, 2001), pp. 19–40
[8] R. Canetti, U. Feige, O. Goldreich, M. Naor, Adaptively secure multiparty computation, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing* (1996), pp. 639–648
[9] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, Universally composable two-party and multi-party computation, in *34th STOC* (2002), pp. 494–503. Full version available at http://eprint.iacr.org/2002/140
[10] Y. Chang, C. Hsiao, C. Lu, On the impossibilities of basing one-way permutations on central cryptographic primitives, in *ASIACRYPT 2002* (2002), pp. 110–124
[11] J.S. Coron, J. Patarin, Y. Seurin, The random oracle model and the ideal cipher model are equivalent, in *CRYPTO 2008*. LNCS, vol. 5157 (Springer, Berlin, 2008), pp. 1–20
[12] I. Damgård, On the existence of bit commitment schemes and zero-knowledge proofs, in *Proc. CRYPTO '89* (1989), pp. 17–27
[13] I. Damgård, Interactive hashing can simplify zero-knowledge protocol design without computational assumptions, in *Proc. CRYPTO '93* (1993), pp. 100–109
[14] I. Damgård, J.B. Nielsen, Improved non-committing encryption schemes based on a general complexity assumption, in *CRYPTO '00* (2000), pp. 432–450
[15] S. Even, O. Goldreich, A. Lempel, A randomized protocol for signing contracts. *Commun. ACM* **28**(6), 637–647 (1985)
[16] U. Feige, A. Shamir, Zero knowledge proofs of knowledge in two rounds, in *CRYPTO'89*. LNCS, vol. 435 (Springer, Berlin, 1989), pp. 526–544
[17] M. Fischlin, On the impossibility of constructing NonInteractive StatisticallySecret protocols from any trapdoor OneWay function, in *Cryptographers' Track—RSA 2002*. LNCS, vol. 2271 (Springer, Berlin, 2002), pp. 79–95
[18] R. Gennaro, Y. Gertner, J. Katz, L. Trevisan, Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput.* **35**(1), 217–246 (2005)
[19] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, M. Viswanathan, The relationship between public key encryption and oblivious transfer, in *The 41st FOCS* (2000), pp. 325–335
[20] Y. Gertner, T. Malkin, O. Reingold, On the impossibility of basing trapdoor functions on trapdoor predicates, in *The* 42*nd FOCS* (2001), pp. 126–135
[21] Y. Gertner, T. Malkin, S. Myers, Towards a separation of semantic and CCA security for public key encryption, in *The 4th TCC*. LNCS, vol. 4392 (Springer, Berlin, 2007), pp. 434–455
[22] O. Goldreich, Kahan, How to construct constant-round zero-knowledge proof systems for NP. *J. Cryptol.* **9**(3), 167–190 (1996)
[23] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications* (Cambridge University Press, Cambridge, 2004)
[24] O. Goldreich, S. Micali, A. Wigderson, Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *J. ACM* **38**(1), 691–729 (1991)
[25] C. Hazay, Y. Lindell, *Efficient Secure Two-Party Protocols* (Springer, Berlin, 2010)
[26] R. Impagliazzo, M. Luby, One-way functions are essential for complexity based cryptography, in *The 30th FOCS* (1989), pp. 230–235
[27] R. Impagliazzo, S. Rudich, Limits on the provable consequences of one-way permutations, in *21st STOC* (1989), pp. 44–61
[28] T. Itoh, Y. Ohta, H. Shizuya, A language-dependent cryptographic primitive. *J. Cryptol.* **10**(1), 37–49 (1997)
[29] B. Kapron, L. Malka, V. Srinivasan, A characterization of non-interactive instance-dependent commitment-schemes (NIC), in *Proc. ICALP 2007* (2007), pp. 328–339

[30] J. Kahn, M. Saks, C. Smyth, A dual version of Reimer's inequality and a proof of Rudich's conjecture, in *Proceedings of the 15th Annual IEEE Conference on Computational Complexity*, July 04–07, 2000, p. 98

[31] J.H. Kim, D.R. Simon, P. Tetali, Limits on the efficiency of one-way permutation-based hash functions, in *The 40th FOCS* (1999), pp. 535–542

[32] M. Luby, C. Rackoff, How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.* **17**(2), 373–386 (1988)

[33] D. Micciancio, S. Vadhan, Statistical zero-knowledge proofs with efficient provers: lattice problems and more, in *CRYPTO 2003*. LNCS, vol. 2729 (Springer, Berlin, 2003), pp. 282–298

[34] D. Micciancio, S.J. Ong, A. Sahai, S. Vadhan, Concurrent zero knowledge without complexity assumptions, in *TCC 2006*. LNCS, vol. 3876 (Springer, Berlin, 2006), pp. 1–20

[35] M. Naor, Bit commitment using pseudorandomness. *J. Cryptol.* **4**(2), 151–158 (1991)

[36] M.H. Nguyen, S. Vadhan, Zero knowledge with ecient provers, in *Proc. 38th STOC* (2006), pp. 287–295

[37] S.J. Ong, S. Vadhan, An equivalence between zero knowledge and commitments, in *The 5th TCC*. LNCS, vol. 4948 (Springer, Berlin, 2008), pp. 482–500

[38] S. Rudich, The use of interaction in public cryptosystems (Extended Abstract), in *CRYPTO 91* (1991), pp. 242–251

[39] O. Reingold, L. Trevisan, S.P. Vadhan, Notions of reducibility between cryptographic primitives, in *The 1st TCC*. LNCS, vol. 2951 (Springer, Berlin, 2004), pp. 1–20

[40] D.R. Simon, Finding collisions on a one-way street: Can secure Hash functions be based on general assumptions? in *EUROCRYPT 1998*. LNCS, vol. 1403 (Springer, Berlin, 1998), pp. 334–345

[41] S.P. Vadhan, An unconditional study of computational zero knowledge, in *The 45th FOCS* (2004), pp. 176–185