Adaptive Zero-Knowledge Proofs and Adaptively Secure Oblivious Transfer

Yehuda Lindell and Hila Zarosim

Department of Computer Science
Bar-Ilan University, Israel
{zarosih,lindell}@cs.biu.ac.il

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). 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 securely 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.

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

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-00457-5_36

O. Reingold (Ed.): TCC 2009, LNCS 5444, pp. 183–201, 2009.

"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 [15] (and all others known) 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 $\mathcal{NP}$ 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 [7,8]). However, the question of whether or not adaptive zero-knowledge *proofs* exist for all $\mathcal{NP}$ 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 [16]. 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 [10,14], all constructions for adaptively secure oblivious transfer use additional assumptions like the ability to sample a permutation without knowing its trapdoor [2,8].

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, this means that the protocol has only computational soundness, because an all-powerful prover is able to equivocate like the simulator. 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* $\mathcal{NP}$.

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 [3,18]. 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 [21,26,22]. 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 [11]and [8]) we

[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.

are able to construct the first computational zero-knowledge *proof* for all $\mathcal{NP}$ that is secure also in the case of adaptive corruptions.[2]

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 [10], 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 [17] and later used in [25,19,12,24] 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 [10,14], 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

[2] In [22], adaptively secure commitment schemes are constructed for the languages of Graph Isomorphism and Quadratic Residuosity (although they are not presented in this way nor for this purpose). The constructions in [22] 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 $\mathcal{SZK}$).

encryption oracles. We prove this using the recent result of [9] 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 [20]). This enables us to extend the black-box separation of [17] to show that key agreement does not exist relative to most symmetric encryption oracles. We conclude the proof by recalling that key agreement can be constructed from oblivious transfer [12]. 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 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 adaptively.

Organization. Due to lack of space in this extended abstract, we present only proof sketches of our results; the full proofs can be found in the full version. Likewise, we do not present the definitions of secure computation and refer to [14] for the definition of security in the presence of semi-honest static adversaries (needed in Section 3.2), and to [6] for the definition of security in the presence of adaptive adversaries. Very briefly, the formulation of security for adaptive adversaries in [6] includes an environment $\mathcal{Z}$ that communicates with the adversary before and after the execution. An important property of the definition is that of *post-execution corruption*, meaning that the environment can ask the adversary to corrupt parties after the protocol transcript has already been fixed. This property is crucial for proving sequential composition; see [6].

2 Adaptive Zero-Knowledge Proofs

In this section we show how to construct 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 G, and the verifier then chooses randomly whether to check 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. Note 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 – given an appropriate trapdoor – can "explain" commitments to 1 as commitments to 0 and vice versa (actually it suffices that commitments to 0 be explainable as commitments to 1). 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 adaptive property of the commitment scheme to fool the verifier).

We overcome this problem and construct adaptive zero-knowledge *proofs* for all $\mathcal{NP}$ by constructing an *adaptive* instance-dependent commitment scheme. Informally speaking, an adaptive instance-dependent commitment scheme (AIDCS) for an NP-language L is comprised of 3 algorithms: **(1)** An ordinary commitment algorithm Com that on an instance x, a bit b, and random coins r returns a commitment c to b, denoted $\mathsf{Com}(x, b; r)$; **(2)** A "fake" commitment algorithm Com' that on an instance x and random coins r' returns a commitment $c' = \mathsf{Com}'(x; r')$; **(3)** An "adaptive-opening" algorithm Adapt that given $x \in L$ and a witness $w \in R_x$, can present every output c' of Com' as a valid commitment to any bit b. That is, given c', the random coins r' used by Com' to generate c' and any bit b, algorithm Adapt outputs "random coins" r such that $c' = \mathsf{Com}(x, b; r)$. Note the difference between Com and Com': While Com is an ordinary committing algorithm (creating a commitment value for a given bit), when $x \in L$, 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.

The security requirements of $(\mathsf{Com}, \mathsf{Com}', \mathsf{Adapt})$ are as follows. For every $x \in L$, the commitment scheme must be hiding, meaning that commitments to 0, to 1 and fake commitments are all indistinguishable (i.e., $\{\mathsf{Com}(x, 0)\}_{x \in L} \overset{\text{c}}{\equiv} \{\mathsf{Com}(x, 1)\}_{x \in L} \overset{\text{c}}{\equiv} \{\mathsf{Com}'(x)\}_{x \in L}$). Furthermore, the output of Adapt must be indistinguishable from the output of an "honest committer" using algorithm Com. More specifically, for every c' in the range of Com' and every bit b, when given a witness $w \in R_x$ and coins r' such that $c' = \mathsf{Com}'(x; r')$, algorithm Adapt outputs a string r that is computationally indistinguishable from a uniformly distributed string and for which $c' = \mathsf{Com}(x, b; r)$. In addition to the above hiding properties we require that for every $x \notin L$, the commitment scheme Com is *perfectly* binding (i.e., $\{\mathsf{Com}(x, 0)\}_{x \notin L} \cap \{\mathsf{Com}(x, 1)\}_{x \notin L} = \phi$). A formal definition appears in the full version of this paper.

Constructing adaptive instance-dependent schemes. Our construction is almost identical to the trapdoor commitment scheme of [11] (as adapted by [8]), with one small but crucial difference. We begin by describing the adaptation by [8] of the trapdoor commitment of [11]. Let C be a perfectly binding commitment scheme with pseudorandom range and let G be a graph (in [11] G is a Hamiltonian graph generated by the receiver whereas in [8] 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 this is indistinguishable from a commitment to 0). 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 any 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. This is due to the fact that when a cycle is known in G, it is possible to decommit to the cycle only (and claim that the rest of the commitments are just random coins), or to decommit to the entire graph.

In summary, we construct the following tuple of probabilistic polynomial-time algorithms: Com works as described above. Algorithm Com' simply generates a commitment to 0 (that is, $\mathsf{Com}'(G;r) = \mathsf{Com}(G,0;r)$). Given a witness $w \in R_G$ (a Hamiltonian cycle in G) and a commitment c in the range of Com', if Adapt has to explain c as a commitment to 0, then it simply outputs the random coins used by Com'. In contrast, in order to explain c as a commitment to 1, Adapt outputs the randomness used by C for the edges in the Hamiltonian cycle in $\pi(G)$ (recall Adapt receives a Hamiltonian cycle w in G as input) and simply claims that all the other commitments are merely random strings (recall that C has pseudorandom range and therefore the output of Adapt is computationally indistinguishable from the uniform distribution).

Adaptive zero-knowledge proof for Hamiltonicity. Our adaptive ZK proof system is exactly that of [5], with the ordinary commitment scheme replaced by an adaptive instance-dependent commitment scheme. The fact that this scheme has unconditional soundness follows from the fact that when $x \notin L$, the commitment is perfectly binding. The simulation works like the standard zero-knowledge simulator for Hamiltonicity except that Com' is used to commit to all edges outside of the n-cycle (in the case that the simulator sends a graph containing only a cycle). This enables the simulator to use Adapt later in case the prover is corrupted, and show that the commitment was "really" to a permutation of G.

3 Adaptive Oblivious Transfer

We prove our black-box separation of adaptively secure oblivious transfer from enhanced trapdoor permutations in the following steps. First, in Section 3.1 we define Γ and Δ oracles, where a Γ-oracle essentially represents an enhanced trapdoor permutation and a Δ-oracle is essentially a type of symmetric encryption scheme. Then, in Section 3.2 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 for measure 1 of random Δ-oracles no statically secure OT_1^2 exists. This is done by using the original black-box separation of key agreement from one-way functions [17], and the fact that key agreement can be obtained from statically secure oblivious transfer; see Section 3.3. We conclude that for measure 1 of random Γ-oracles no adaptively secure OT_1^2 exists (see Section 3.4).

3.1 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 all over $\{0,1\}^n$ 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:

- $G_\Gamma(\cdot) = (G_\Gamma^1, G_\Gamma^2)$ is a pair of injective functions such that on an input $r \in \{0,1\}^n$, $G_\Gamma(r) = (G_\Gamma^1(r), G_\Gamma^2(r)) = (fid, tid) \in \{0,1\}^{2n} \times \{0,1\}^{2n}$. Note that a party can query only G_Γ and cannot query one of its components separately.
- A function $F(\cdot,\cdot)$, such that for every $fid \in Range(G_\Gamma^1)$, $F(fid,\cdot)$ is a permutation over $\{0,1\}^n$ and for every $fid \notin Range(G_\Gamma^1)$ 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 $(fid,tid) \in Range(G_\Gamma)$. If tid is not in the range of $G_\Gamma^2(\cdot)$, then F^{-1} returns $\perp$. Note that since G_Γ^1 and G_Γ^2 are injective functions, pairs of the form (fid,tid) and (fid',tid), where $fid \neq fid'$ do not exist and F^{-1} is well defined.

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^1 and G_Γ^2 are two uniformly distributed injective functions from $\{0,1\}^n$ to $\{0,1\}^{2n}$ and for every $fid \in Range(G_\Gamma^1)$, $F(fid,\cdot)$ 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. For reasons that will become apparent later, we also define a function Q and its inverse (this has no analogue 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 $x \in \{0,1\}^n$, $P^{-1}(fid,x) = \perp$.
- Q is an injective function from the range of G_Δ to $\{0,1\}^{2n}$. Namely, for every $fid \in Range(G_\Delta)$, $Q(fid) \in \{0,1\}^{2n}$, for every $fid \neq fid' \in Range(G_\Delta)$, $Q(fid) \neq Q(fid')$ and for every $fid \notin Range(G_\Delta)$, $Q(fid) = \perp$.
- Q^{-1} is the inversion algorithm of Q. Namely, for every $fid \in Range(G_\Delta)$, $Q^{-1}(Q(fid)) = fid$. for every $y \notin Range(Q)$, $Q^{-1}(y) = \perp$.

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

Note the difference between Γ-oracles and Δ-oracles. Γ-oracle 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 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 $(G_\Delta, P, P^{-1}, Q, Q^{-1})$ satisfying:

- For every $r \in \{0,1\}^n$, it holds that $G_\Delta(r) = G_\Gamma^1(r)$.
- For every $r \in \{0,1\}^n$, $Q(G_\Delta(r)) = G_\Gamma^2(r)$, and for every $fid \notin Range(G_\Delta)$, $Q(fid) = \perp$.
- For every $fid \in \{0,1\}^{2n}$ and $x \in \{0,1\}^n$, it holds that $P(fid,x) = F(fid,x)$.
- P^{-1} and Q^{-1} are the inversion algorithms of P and Q.

Claim 1. *ϕ is a bijection from the set of Γ-oracles to the set of Δ-oracles.*

The above claim is proven in the full version of this paper and immediately implies the following:

Corollary 2. *The random variables $\mathcal{U}_{\Delta}$ and $\phi(\mathcal{U}_{\Gamma})$ are identically distributed.*

Enhanced trapdoor permutations relative to Γ-oracles. It is not difficult to show there exist enhanced trapdoor permutations, as defined in [14], relative to random Γ-oracles. Indeed, it can be shown that there exist enhanced trapdoor permutations relative to measure 1 of the Γ-oracles. This is shown in the full version. We remark also that semi-honest oblivious transfer with static corruptions can be constructed from any enhanced trapdoor permutation [10] and thus exists relative to measure 1 of the Γ-oracles.

3.2 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 a 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 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.

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. 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 the probability of a party finding an fid in the range of G_{Γ} without explicitly querying it is negligible. 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 r to the oracle G_Γ, $\mathcal{P}_2$ first queries $G_\Delta(r)$ and gets an output fid. Then, $\mathcal{P}_2$ queries $Q(fid)$ and gets an output tid. $\mathcal{P}_2$ hands the pair (fid, tid) 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 Q^{-1} on tid and receives an output fid. If the outputs is $\perp$, it hands $\perp$ to $\mathcal{P}_1$. Otherwise, $\mathcal{P}_2$ 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 3. *If Π_1 securely computes the 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 Sketch: 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 show that Π_2 is a statically secure protocol relative to O_Δ. We use the ideal-process simulator $\mathcal{SIM}$ of Π_1 for the adaptive setting to construct two probabilistic polynomial-time simulators $\mathcal{S}_{Alice_2}$ and $\mathcal{S}_{Bob_2}$ for Π_2 in the static setting. Due to space restrictions, we present below only $\mathcal{S}_{Bob_2}$ (the simulator $\mathcal{S}_{Alice_2}$ is almost identical). 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 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 $G_\Delta(r)$ and receives an output fid. It then queries Q on fid, gets an output tid and hands the pair (fid, tid) 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 Q^{-1} on tid, gets an output fid. If the outputs is $\perp$, it hands $\perp$ to $\mathcal{SIM}$. Otherwise, $\mathcal{S}_{Bob_2}$ 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 $\{0,1\}^{2n} \times \{0,1\}^{2n}$, 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, the distinguisher D_1 begins by initializing a table T_Q 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) and for every access of one of the parties to G_Γ, namely a query $G_\Gamma(r) = (fid, tid)$, D_1 records the entry (fid, tid) in T_Q. D_1 starts simulating D_2 on the view of Bob and proceeds as follows:

- If D_2 makes a query $G_\Delta(r)$, D_1 makes a query $G_\Gamma(r)$, gets a pair (fid, tid), records the entry (fid, tid) in T_Q and hands fid to D_2.
- If D_2 tries to compute $Q(fid)$, D_1 looks for an entry (fid, tid) in T_Q. If such an entry exists, it hands tid to D_2 and continues. Else, it hands $\perp$ to D_2.
- If D_2 tries to compute $Q^{-1}(tid)$, D_1 looks for (fid, tid) in T_Q. If such an entry exists, it hands fid to D_2 and continues. Otherwise, it hands $\perp$ 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_Q. If not, it returns $\perp$. If yes, it queries $F^{-1}(tid, y)$ and returns its answer.

There are two cases: If the simulated D_2 does not make a query on an fid (or its corresponding tid) in the range of G_Δ that does not appear in T_Q, then the run of D_1 with O_Γ is identical to a run of D_2 with O_Δ and therefore D_1 outputs the same as D_2. On the other hand, if the simulated D_2 does make such a query, then the output of D_1 might be different than that of D_1 (since, for such queries D_1 replies by $\perp$, while the real O_Δ's reply is different). However, D_2 can find such an fid (or tid) with only negligible probability and therefore if D_2 is a distinguisher for Π_2, D_1 is a distinguisher for Π_1. ■

Remark 4. *Theorem 3 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.*

Needless to say, Theorem 3 holds for oblivious transfer as a special case.

3.3 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 [17] 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 $\{0,1\}^{2n}$.

- G_{TEST} is a function that returns a string in $\{0,1\}^n$ on inputs in the range of $G_\rho(\cdot)$. For any other input, it returns $\perp$. Note that G_{TEST} is in fact a tool for examining whether a string of size $2n$ is in the range of G_ρ or not.[3]
- F_P is a function that on a triple $(I,k,x) \in \{0,\ldots,5\} \times \{0,1\}^{2n} \times \{0,1\}^{\frac{n}{2}}$ returns a string $y \in \{0,1\}^{\frac{n}{2}}$. Note that for a given I and $k \in \{0,1\}^{2n}$, $F_P(I,k,\cdot)$ is a function from $\{0,1\}^{\frac{n}{2}}$ to $\{0,1\}^{\frac{n}{2}}$.
- F_Q is a function that on a pair $(I,x) \in \{0,\ldots,5\} \times \{0,1\}^n$ returns a string $y \in \{0,1\}^n$. Thus, for a given I, $F_Q(I,\cdot)$ is a function from $\{0,1\}^n$ to $\{0,1\}^n$.

Note that the output of G_ρ is an fid – or symmetric key k – of length $2n$ which defines 6 random functions $F_P(0,k,\cdot),\ldots,F_P(5,k,\cdot)$ which are then used to simulate the P permutation of a Δ-oracle, using Luby-Rackoff. Likewise, the index I in F_Q is used for deriving 6 different function for Luby-Rackoff (there is no "secret key" k for F_Q because it is used for simulating the Q permutation in a Δ oracle which is not keyed).

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

Δ-oracles versus ρ-oracles. We now use the Luby-Rackoff construction [20] to replace a random Δ-oracle with a random ρ-oracle. We stress that unlike Corollary 2, the distributions are only computationally indistinguishable.

Definition 5 (Feistel Permutation). *Let $f : \{0,1\}^l \to \{0,1\}^l$ be a function and let $x_1, x_2 \in \{0,1\}^l$. DES_f is the permutation defined by $DES_f(x_1,x_2) \stackrel{\text{def}}{=} (x_2, x_1 \oplus f(x_2))$. $DES_{f_1,\ldots,f_k}$ is the permutation defined by $DES_{f_1,\ldots,f_k}(x_1,x_2) \stackrel{\text{def}}{=} DES_{f_2,\ldots,x_k}(DES_{f_1}(x_1,x_2))$.*

Note that inverting a Feistel permutation is no harder than computing it, as $DES_f^{-1}(y_1,y_2) = (y_2 \oplus f(y_1), y_1)$. Intuitively, a Feistel permutation upon a random ρ-oracle can be used in order to obtain an oracle that behaves like a Δ-oracle. Formally, for a given ρ-oracle $O_\rho = (G_\rho, G_{TEST}, F_P, F_Q)$, an $fid \in \{0,1\}^{2n}$ and $x_1, x_2 \in \{0,1\}^{\frac{n}{2}}$, we define six functions: $f_0 = F_P(0,fid,\cdot)$, $f_1 = F_P(1,fid,\cdot)$, $f_2 = F_P(2,fid,\cdot)$, $f_3 = F_P(3,fid,\cdot)$,$f_4 = F_P(4,fid,\cdot)$ and $f_5 = F_P(5,fid,\cdot)$. Then, the permutation $PDES$ relative to a given oracle O_ρ, that simulates the P permutation in the Δ-oracle, is defined by

$$PDES_{O_\rho,fid}(x_1,x_2) \stackrel{\text{def}}{=} DES_{f_0,\ldots,f_5}(x_1,x_2)$$

[3] It was shown in [12] that the black-box separation of [17] holds when G_{TEST} is added to the oracle defined in [17].

Note that $PDES_{O_\rho,fid}$ is a permutation over $\{0,1\}^n$ (similar to $P(fid,\cdot)$ in a Δ-oracle). Let $PDES^{-1}_{O_\rho,fid}$ be the inverse permutation. Similarly, for a given ρ-oracle $O_\rho = (G_\rho, G_{TEST}, F_P, F_Q)$ and for $x_1, x_2 \in \{0,1\}^n$ we define $g_0 = F_Q(0,\cdot), \ldots, g_5 = F_Q(5,\cdot)$. (Recall that Q oracle queries in a Δ-oracle are not keyed and thus when simulated using F_Q in a ρ-oracle, no key is used.) We define:

$$QDES_{O_\rho}(x_1,x_2) \stackrel{\text{def}}{=} DES_{g_0,\ldots,g_5}(x_1,x_2)$$

As above, $QDES_{O_\rho}$ is a permutation over $\{0,1\}^{2n}$ (similar to Q in a Δ-oracle). Let $QDES^{-1}_{O_\rho}$ be the inverse permutation.

We define a mapping ψ from ρ to Δ oracles. Let $O_\rho = (G_\rho, G_{TEST}, F_P, F_Q)$ be a ρ-oracle. Then $\psi(O_\rho) = (G_\Delta, P, P^{-1}, Q, Q^{-1})$ is the following Δ-oracle:

- For every $r \in \{0,1\}^n$, $G_\Delta(r) = G_\rho(r)$
- For every $fid \in Range(G_\Delta)$ and all $x \in \{0,1\}^n$, $P(fid,x) = PDES_{O_\rho,fid}(x)$
- For every $fid \notin Range(G_\Delta)$ and for every $x \in \{0,1\}^n$, $P(fid,x) = \perp$
- For every $fid \in Range(G_\Delta)$, $Q(fid) = QDES_{O_\rho}(fid)$
- For every $fid \notin Range(G_\Delta)$, $Q(fid) = \perp$
- P^{-1} and Q^{-1} are the inverse functions of P and Q

We denote by $\psi(\mathcal{U}_\rho)$ the distribution where a random ρ-oracle is chosen and then ψ is applied to it. The following claim states that access to a random Δ-oracle O_Δ is essentially the same as access to a Δ-oracle $\psi(O_\rho)$, when O_ρ is random.

Theorem 6 ([9]). *There exists a simulator $\mathcal{S}$ and a negligible function μ, such that for every machine D with unbounded running time which makes a polynomial number of queries,*

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

We remark that [9] refer to a plain random oracle and a plain random permutation, without the additional fid generating and other functions. However, $G_\rho = G_\Delta$ by definition, and so clearly G_ρ can be simulated given G_Δ. Likewise, G_{TEST} can be simulated using P (because the latter returns $\perp$ if the fid is not in the range). We use Theorem 6 in order to prove the following theorem:

Theorem 7. *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^2_1 functionality.*

In order to prove Theorem 7, we recall the original black-box separation of key agreement from a random oracle, as proven in [17].

Theorem 8 ([17]). *If $\mathcal{P} = \mathcal{NP}$, then given any key-agreement protocol relative to a random ρ-oracle*[4]*, for every polynomial $poly(\cdot)$, there exists a polynomial time Eve such that Eve finds all intersection queries with probability $1 - \frac{1}{poly(n)}$.*

[4] [17] refer to a single random permutation oracle; however, the same proof can be extended to ρ-oracles.

We first show that a similar argument holds relative to Δ-oracles (that is, every key agreement protocol relative to a random Δ-oracle can be broken with probability $1-\frac{1}{poly(n)}$). Then, using the same methods as in [17], we show that relative to measure 1 of Δ-oracles, any key-agreement can be broken in polynomial time. As described in [12], 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. We begin by proving the following claim:

Proposition 9. *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 $1 - \frac{1}{2poly(n)}$.*

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 while replacing queries to P, P^{-1}, Q and Q^{-1} by appropriate Feistel permutations obtained from F_P and F_Q.

Let $\langle \mathcal{A}_2, \mathcal{B}_2 \rangle$ be the following protocol:

Protocol 1. *On input 1^n, $\mathcal{A}_2$ invokes $\mathcal{A}_1$ on 1^n and $\mathcal{B}_2$ invokes $\mathcal{B}_1$ on 1^n. The execution is described below for a party $\mathcal{P}_2$ emulating $\mathcal{P}_1$, and is the same for both $\mathcal{A}_2$ and $\mathcal{B}_2$. In each round:*

- *When $\mathcal{P}_2$ gets the message sent by the other party in the previous round, it sends it to $\mathcal{P}_1$.*
- *If $\mathcal{P}_1$ makes a query r to oracle G_Δ, $\mathcal{P}_1$ queries it oracle $G_\rho(r)$, and hands the output to $\mathcal{P}_1$.*
- *If $\mathcal{P}_1$ makes a query $P(fid, x)$, $\mathcal{P}_1$ queries its oracle G_{TEST} on fid (recall that $G_{TEST}(fid)$ returns $\perp$ if and only if fid is not in the range of G_ρ). If the oracle returns $\perp$, $\mathcal{P}_1$ returns $\perp$ as well. Otherwise, uses its oracle F_P to compute $y = PDES_{O_\rho, fid}(x)$ and hands y to $\mathcal{P}_1$.*
- *If $\mathcal{P}_1$ makes a query $P^{-1}(fid, y)$, $\mathcal{P}_1$ queries G_{TEST} on fid. If it returns $\perp$, $\mathcal{P}_1$ returns $\perp$ as well. Otherwise, $\mathcal{P}_1$ uses its oracle F_P to compute $x = PDES^{-1}_{O_\rho, fid}(y)$ and hands x to $\mathcal{P}_1$.*
- *If $\mathcal{P}_1$ makes a query $Q(fid)$, $\mathcal{P}_1$ queries G_{TEST} on fid. If it returns $\perp$, $\mathcal{P}_1$ returns $\perp$ as well. Otherwise, it uses its oracle F_Q to compute $tid = QDES_{O_\rho}(fid)$ and hands tid to $\mathcal{P}_1$.*
- *If $\mathcal{P}_1$ makes a query $Q^{-1}(tid)$, $\mathcal{P}_1$ uses its oracle F_Q to compute $fid = QDES^{-1}(tid)$ and queries $G_{TEST}(fid)$. If it returns $\perp$, $\mathcal{P}_1$ returns $\perp$ as well. Otherwise, $\mathcal{P}_1$ hands fid to $\mathcal{P}_1$.*
- *If $\mathcal{P}_1$ writes a string m on its outgoing communication tape, $\mathcal{P}_1$ sends m to the other party.*
- *At the end of the protocol, $\mathcal{P}_1$ outputs the output of $\mathcal{P}_1$.*

Now, assume $\mathcal{P} = \mathcal{NP}$. Let $poly(\cdot)$ be some polynomial and let Eve_2 be as in Theorem 8. 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 6. Note that if Eve_1 outputs a list of intersection queries with probability less than $1 - \frac{1}{2poly(n)}$, then it is possible to distinguish oracles $\mathcal{U}_\rho, \psi(\mathcal{U}_\rho)$ 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 $\mathcal{U}_\rho, \psi(\mathcal{U}_\rho)$ 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 $\mathcal{U}_\rho, \psi(\mathcal{U}_\rho)$ then Eve_2 outputs all intersection queries with probability at least $1 - \frac{1}{poly(n)}$, 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 $1 - \frac{1}{2poly(n)}$. Thus D distinguishes with non-negligible probability. ∎

Remark 10. *Theorem 6 holds even when $\mathcal{P} = \mathcal{NP}$ since the running time of D is unbounded.*

The following corollary can be proved using the same methods as in [17] (the only difference between it and what was proven in [17] is the type of oracle used):

Corollary 11. *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 12. *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.*

3.4 Concluding the Proof

Theorem 3 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 7, for measure 1 of Δ oracles, there exists no statically secure OT_1^2. Using the fact that ϕ is a bijection (Claim 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 13. *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 to [17], we derive an oracle separation of enhanced trapdoor permutations form adaptively secure OT_1^2 (even for semi-honest adversaries):

Corollary 14. *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. Beaver, D.: Adaptive Zero Knowledge and Computational Equivocation. In: 28th STOC, pp. 629–638 (1996)
2. Beaver, D.: Adaptively Secure Oblivious Transfer. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 300–314. Springer, Heidelberg (1998)
3. Bellare, M., Micali, S., Ostrovsky, R.: Perfect Zero-Knowledge in Constant Rounds. In: 22nd STOC, pp. 482–493 (1990)
4. Blum, M.: Coin Flipping by Phone. In: IEEE Spring COMPCOM, pp. 133–137 (1982)
5. Blum, M.: 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. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology 13(1), 143–202 (2000)
7. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)
8. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally Composable Two-Party and Multi-Party Computation. In: 34th STOC, pp. 494–503 (2002), http://eprint.iacr.org/2002/140
9. Coron, J.S., Patarin, J., Seurin, Y.: The Random Oracle Model and the Ideal Cipher Model are Equivalent. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 1–20. Springer, Heidelberg (2008)
10. Even, S., Goldreich, O., Lempel, A.: A Randomized Protocol for Signing Contracts. Communications of the ACM 28(6), 637–647 (1985)
11. Feige, U., Shamir, A.: Zero Knowledge Proofs of Knowledge in Two Rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, Heidelberg (1990)
12. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The Relationship Between Public Key Encryption and Oblivious Transfer. In: 41st FOCS, pp. 325–335 (2000)
13. Gertner, Y., Malkin, T., Reingold, O.: On the Impossibility of Basing Trapdoor Functions on Trapdoor Predicates. In: The 42nd FOCS, pp. 126–135 (2001)
14. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
15. Goldreich, O., Micali, S., Wigderson, A.: Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. Journal of the ACM 38(1), 691–729 (1991)
16. Impagliazzo, R., Luby, M.: One-way Functions are Essential for Complexity Based Cryptography. In: The 30th FOCS, pp. 230–235 (1989)
17. Impagliazzo, R., Rudich, S.: Limits on the Provable Consequences of One-way Permutations. In: 21st STOC, pp. 44–61 (1989)
18. Itoh, T., Ohta, Y., Shizuya, H.: A Language-Dependent Cryptographic Primitive. Journal of Cryptology 10(1), 37–49 (1997)
19. Kim, J.H., Simon, D.R., Tetali, P.: Limits on the Efficiency of One-Way Permutation-Based Hash Functions. In: The 40th FOCS, pp. 535–542 (1999)
20. Luby, M., Rackoff, C.: How to Construct Pseudorandom Permutations from Pseudorandom Functions. SIAM Journal on Computing 17(2), 373–386 (1988)

21. Micciancio, D., Vadhan, S.: Statistical Zero-Knowledge Proofs with Efficient Provers: Lattice Problems and More. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 282–298. Springer, Heidelberg (2003)
22. Micciancio, D., Ong, S.J., Sahai, A., Vadhan, S.: Concurrent Zero Knowledge without Complexity Assumptions. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 1–20. Springer, Heidelberg (2006)
23. Naor, M.: Bit Commitment Using Pseudorandomness. Journal of Cryptology 4(2), 151–158 (1991)
24. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of Reducibility between Cryptographic Primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
25. Simon, D.R.: Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
26. Vadhan, S.P.: An Unconditional Study of Computational Zero Knowledge. In: The 45th FOCS, pp. 176–185 (2004)