Injective Trapdoor Functions via Derandomization: How Strong is Rudich's Black-Box Barrier?

Lior Rotem* Gil Segev*

Abstract

We present a cryptographic primitive $\mathcal{P}$ satisfying the following properties:

- Rudich's seminal impossibility result (PhD thesis '88) shows that $\mathcal{P}$ cannot be used in a black-box manner to construct an injective one-way function.
- $\mathcal{P}$ can be used in a *non-black-box* manner to construct an injective one-way function assuming the existence of a hitting-set generator that fools deterministic circuits (such a generator is known to exist based on the *worst-case* assumption that $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ has a function of deterministic circuit complexity $2^{\Omega(n)}$).
- Augmenting $\mathcal{P}$ with a trapdoor algorithm enables a non-black-box construction of an injective *trapdoor* function (once again, assuming the existence of a hitting-set generator that fools deterministic circuits), while Rudich's impossibility result still holds.

The primitive $\mathcal{P}$ and its augmented variant can be constructed based on any injective one-way function and on any injective trapdoor function, respectively, and they are thus unconditionally essential for the existence of such functions. Moreover, $\mathcal{P}$ can also be constructed based on various known primitives that are secure against related-key attacks, thus enabling to base the strong structural guarantees of injective one-way functions on the strong security guarantees of such primitives.

Our application of derandomization techniques is inspired mainly by the work of Barak, Ong and Vadhan (CRYPTO '03), which on one hand relies on any one-way function, but on the other hand only results in a non-interactive perfectly-binding commitment scheme (offering significantly weaker structural guarantees compared to injective one-way functions), and does not seem to enable an extension to public-key primitives.

The key observation underlying our approach is that Rudich's impossibility result applies not only to one-way functions as the underlying primitive, but in fact to a variety of "unstructured" primitives. We put forward a condition for identifying such primitives, and then subtly tailor the properties of our primitives such that they are both sufficiently unstructured in order to satisfy this condition, and sufficiently structured in order to yield injective one-way and trapdoor functions. This circumvents the basic approach underlying Rudich's long-standing evidence for the difficulty of constructing injective one-way functions (and, in particular, injective trapdoor functions) based on seemingly weaker or unstructured assumptions.

A preliminary version of this work appeared in *Proceedings of the 16th Theory of Cryptography Conference (TCC)*, 2018.

*School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: `{lior.rotem,segev}@cs.huji.ac.il`. Supported by the European Union's 7th Framework Program (FP7) via a Marie Curie Career Integration Grant (Grant No. 618094), by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253), by the Israel Science Foundation (Grant No. 483/13), by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11), by the US-Israel Binational Science Foundation (Grant No. 2014632), and by a Google Faculty Research Award.

Contents

1 Introduction

Over the last few decades the cryptography community has been successful in constructing a wide variety of cryptographic primitives based on the minimal assumption that one-way functions exist. For example, the existence of one-way functions has been shown equivalent to the existence of private-key encryption schemes [GGM84], pseudorandom functions and permutations [GGM86, LR88, NR99], message authentication codes [GGM86], pseudorandom generators [BM84, HIL^{+}99], universal one-way hash functions and signature schemes [NY89, Rom90], commitment schemes [Nao91, HIL^{+}99, HNO^{+}09], and many other symmetric primitives (also known as "MiniCrypt" primitives [Imp95]).

Despite the great progress in basing symmetric cryptography on one-way functions, the existence of one-way functions is still not known to imply the existence of all symmetric cryptographic primitives. A prime example is that of injective one-way functions (and, in particular, one-way permutations), whose existence seems to require somewhat more structured assumptions (e.g., specific number-theoretic assumptions [GLN11]).[1] Moreover, the seminal work by Rudich [Rud88], within the framework of Impagliazzo and Rudich modeling black-box constructions [IR89, RTV04], provided substantial evidence that the existence of injective one-way functions may not be "naturally implied" by the existence of arbitrary one-way functions. Specifically, Rudich proved that one-way functions cannot be used in a black-box manner to construct injective one-way functions.[2]

Black-box impossibility results are clearly inherently limited, and do not capture non-black-box techniques (e.g., [GMW86, Yao86, NY90, Bar01, AIK06, BP12, CPS16]). Thus, it may still be the case that one-way functions can be used in a non-black-box manner to construct injective one-way functions (and even one-way permutations). Given that Rudich's black-box barrier is currently the main evidence for explaining our lack of success in constructing injective one-way functions based on seemingly weaker assumptions, this naturally raises the fundamental question of whether or not Rudich's black-box barrier can be circumvented using non-black-box techniques.

Significant progress towards obtaining a better understanding of the above question was made in the work of Barak, Ong and Vadhan [BOV07]. Their work demonstrated that derandomization techniques can be fundamentally useful in cryptographic constructions by enabling to eliminate interaction from certain two-message cryptographic protocols. Relying on the existence of a hitting-set generator that fools co-non-deterministic algorithms[3], they derandomized Naor's statistically-binding commitment scheme [Nao91] for obtaining a non-interactive perfectly-binding commitment scheme (in addition, relying on the existence of a hitting-set generator that fools co-non-deterministic circuits, they derandomized Dwork and Naor's ZAPs [DN07] for obtaining a non-interactive witness-indistinguishable proof system for NP).

In particular, as observed by Barak, Ong and Vadhan, a non-interactive perfectly-binding commitment scheme naturally implies a somewhat weak form of an injective one-way function, to which they refer to as a "partially-injective" one-way function. Such a function f is a two-input function $f(x, y)$, which is injective with respect to its first input x but not necessarily with respect to its second input y (thus offers significantly weaker structural guarantees compared to an injective one-way function), and for which it is hard to recover x given $f(x, y)$ where both x and y are distributed uniformly. This shows that non-black-box techniques are useful for constructing a somewhat weak

[1] An additional example is that of collision-resistant hash functions, whose existence also seems to require somewhat stronger assumptions [Sim98].

[2] Although Rudich formalized his statements for one-way permutations, his proof relies only on the injectivity of the resulting functions, and thus applies to injective one-way functions.

[3] Such a generator is known to exist based on the worst-case assumption that $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ has a function that is not computable for infinitely many input lengths by a probabilistic non-deterministic algorithm that runs in sub-exponential time [NW94, IW97, MV99, GSTS03].

form of injective one-way functions, but the problem of whether or not such techniques can be useful for constructing (fully) injective one-way functions (and even trapdoor functions) based on seemingly weaker assumptions has been left completely open.

1.1 Our Contributions

We show that non-black-box techniques can be used to circumvent the basic approach underlying Rudich's long-standing evidence for the difficulty of constructing injective one-way functions (and, in particular, injective trapdoor functions) based on seemingly weaker or unstructured assumptions. In addition, whereas separations between the black-box and non-black-box power of cryptographic constructions were known to exist for private-key primitives [MP12], our work provides in particular such a separation for public-key primitives.

Specifically, we present a cryptographic primitive $\mathcal{P}$ and prove that it satisfies the following properties:

- Rudich's seminal impossibility result shows that $\mathcal{P}$ cannot be used in a black-box manner to construct an injective one-way function.
- $\mathcal{P}$ can be used in a non-black-box manner to construct an injective one-way function assuming the existence of a hitting-set generator that fools deterministic circuits. The non-black-box aspect of our construction is quite modest, asking for an upper bound on the size of $\mathcal{P}$'s implementation.
- Augmenting $\mathcal{P}$ with a trapdoor algorithm enables a non-black-box construction of an injective trapdoor function (once again, assuming the existence of a hitting-set generator that fools deterministic circuits), while Rudich's impossibility result still holds.

Generally speaking, a hitting-set generator that fools deterministic circuits is known to exist based on the worst-case assumption that $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ has a function of deterministic circuit complexity $2^{\Omega(n)}$ (see Section 2.1 for more details). For our construction, however, it suffices to assume the existence of a hitting-set generator that fools a rather simple computation involving the primitive $\mathcal{P}$ (two parallel invocations of $\mathcal{P}$ followed by a comparison of their outputs). Thus, if a hitting-set generator that fools this specific computation is known to exist *unconditionally* then we do not need to rely on the above worst-case assumption.

Our application of derandomization techniques is inspired mainly by the work of Barak, Ong and Vadhan [BOV07], which on one hand relies on any one-way function, but on the other hand only results in a non-interactive perfectly-binding commitment scheme (offering a significantly weaker structural guarantee when compared to injective one-way functions), and does not seem to enable an extension to public-key primitives (see Section 1.3 for an in-depth discussion and comparison to previous applications of derandomization techniques in cryptography).

The primitive $\mathcal{P}$. Our primitive $\mathcal{P}$ is a predicate $\mathcal{P} : \{0,1\}^* \to \{0,1\}$ that satisfies two rather natural properties, and we refer to this primitive as a *correlated-input balanced one-way predicate*. We show that such a predicate $\mathcal{P}$ can be constructed based on any injective one-way function without relying on any additional assumptions, and thus the existence of such a predicate is unconditionally essential for the existence of an injective one-way function. Therefore, under a standard worst-case hardness assumption, the existence of our primitive is equivalent to that of an injective one-way function, although it is strictly weaker when restricted to black-box constructions.

Moreover, we also show that $\mathcal{P}$ can be constructed in a black-box manner from various known primitives that are secure against related-secret attacks (e.g., related-key pseudorandom functions and related-seed pseudorandom generators). Although these primitives seem rather unstructured, it turns out that we can rely on their strong *security guarantees* to achieve the relatively modest *structural guarantee* of $\mathcal{P}$, and then apply derandomization techniques to obtain the more robust structure of injective one-way functions.

In addition to the primitive $\mathcal{P}$, we also introduce a natural "public-key" variant of $\mathcal{P}$ which is obtained by augmenting $\mathcal{P}$ with a trapdoor algorithm. We show that this augmented primitive can be constructed based on any injective trapdoor function without relying on any additional assumptions, and thus the existence of this primitive is unconditionally essential for the existence of an injective trapdoor one-way function. Therefore, similarly to the above, under a standard worst-case hardness assumption, the existence of our augmented primitive is equivalent to that of an injective trapdoor function, although it is strictly weaker when restricted to black-box constructions.

Our approach. The key observation underlying our approach is that Rudich's black-box impossibility result applies not only to rule out black-box constructions of injective one-way functions from general one-way functions as the underlying primitive class, but in fact from a wide variety of "unstructured" primitive classes. As basic examples, these include one-way functions and "almost-injective" one-way functions[4], and obviously do not include injective one-way functions. At a very high level, as we discuss in Section 1.2 in more detail, Rudich's impossibility applies to any primitive class $\mathfrak{S}$ satisfying the following condition: For any $\mathcal{O}, \mathcal{O}' \in \mathfrak{S}$ and for any two *disjoint* sets of inputs X and X' *of polynomial size*, there exists an $\mathcal{O}'' \in \mathfrak{S}$ that agrees with $\mathcal{O}$ on the set X and agrees with $\mathcal{O}'$ on the set X'.

Equipped with this observation, a significant part of our effort in this work focuses on carefully identifying a primitive $\mathcal{P}$ that on one hand is sufficiently unstructured in order to satisfy the above condition, whereas on the other hand it is sufficiently structured in order to yield an injective one-way function (via a non-black-box construction). As we pointed out, one-way functions and almost-injective one-way functions are examples for primitive classes that satisfy the above condition, but it is still a long-standing open problem to use them in order to construct an injective one-way function. Instead, we specifically tailor the properties of our primitive $\mathcal{P}$ in order to simultaneously satisfy the above condition and yield an injective one-way function via derandomization techniques.

1.2 Overview of Our Approach

In this section we provide an overview of our main contributions. First, we describe our new notion of a correlated-input balanced one-way predicate, as well as our non-black-box construction of an injective one-way function. We emphasize that we view the introduction and the specific formalization of our new primitive as a central contribution given that: (1) it is sufficiently unstructured in order to satisfy the above-mentioned condition for Rudich's impossibility result, (2) it is sufficiently structured in order to yield an injective one-way function, and (3) its existence is essential for the existence of an injective one-way function.

Then, we describe the application of Rudich's impossibility proof to correlated-input balanced one-way predicates, and discuss the observation that Rudich's impossibility result applies to a wide variety of primitives. In fact, we prove a stronger result, showing that there is no black-box construction of a *partially-injective* one-way function (as defined by Barak, Ong and Vadhan [BOV07]) from these primitives.

[4]We denote by an "almost-injective" function a function that is injective for each input length on all but a negligible fraction of its domain.

Correlated-input balanced one-way predicates. The new primitive at the heart of our approach is an efficiently-computable predicate $\mathcal{P} : \{0,1\}^* \to \{0,1\}$ that can be viewed as a two-input predicate $\mathcal{P}(x, r)$, where $r \in \{0,1\}^{\ell(|x|)}$, which satisfies the following two natural requirements with respect to correlated inputs:

- The first requirement is that the predicate $\mathcal{P}$ has to be rather balanced in the sense that $|\Pr[\mathcal{P}(x,r) = \mathcal{P}(x',r)] - 1/2|$ is bounded for every distinct $x, x' \in \{0,1\}^n$, where the probability is taken over the choice of a uniform $r \in \{0,1\}^{\ell(n)}$.

  This requirement (on its own) is easy to satisfy by making sure that $\mathcal{P}$ is pair-wise independent over the choice of $r \in \{0,1\}^{\ell(n)}$. For example, this requirement can be satisfied by defining $\mathcal{P}(x,r) = \langle f(x), r\rangle$, where f may be any *injective* function mapping n-bit inputs to $\ell(n)$-bit outputs.

- The second requirement is that for adversarially-chosen values $r_1, \ldots, r_T \in \{0,1\}^{\ell(n)}$, the function mapping x to the sequence of values $\mathcal{P}(x, r_1), \ldots, \mathcal{P}(x, r_T)$ is a one-way function of x.

  This requirement (on its own) is easy to satisfy by making sure that $\mathcal{P}$ first applies any given one-way function to its first input x, and only then involves its second input r in the computation. For example, this requirement can be satisfied by defining $\mathcal{P}(x,r) = \langle f(x), r\rangle$, where f may be any *one-way* function mapping n-bit inputs to $\ell(n)$-bit outputs (note that this predicate fails to satisfy the first requirement whenever f is not an injective function).

The following definition formalizes these two requirements:

Definition 1.1. Let $\mathcal{P} : \{0,1\}^* \to \{0,1\}$ be an efficiently-computable predicate, and let $\ell = \ell(n)$ and $\delta = \delta(n)$ be functions of the security parameter $n \in \mathbb{N}$. Then, $\mathcal{P}$ is a *correlated-input* (ℓ, δ)*-balanced one-way predicate* if it satisfies the following two requirements:

- For any $n \in \mathbb{N}$ and for any $x, x' \in \{0,1\}^n$ such that $x \neq x'$ it holds that

$$\left| \Pr_{r \leftarrow \{0,1\}^{\ell(n)}} \left[\mathcal{P}(x,r) = \mathcal{P}(x',r)\right] - \frac{1}{2} \right| \leq \delta(n).$$

- For any probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\cdot)$ such that

$$\Pr\left[\mathsf{Invert}_{\mathcal{P},A}(n) = 1\right] \leq \nu(n)$$

  for all sufficiently large $n \in \mathbb{N}$, where the experiment $\mathsf{Invert}_{\mathcal{P},A}(n)$ is defined as follows:

  1. $(\mathsf{state}, r_1, \ldots, r_T) \leftarrow A(1^n)$ for $r_1, \ldots, r_T \in \{0,1\}^{\ell(n)}$, where $T = T(n)$ may be any polynomial determined by A.
  2. $x' \leftarrow A\left(\mathsf{state}, \mathcal{P}\left(x, r_1\right), \ldots, \mathcal{P}\left(x, r_T\right)\right)$ where $x \leftarrow \{0,1\}^n$.
  3. If $x' = x$ then output 1, and otherwise output 0.

As demonstrated above, each of the two requirements on its own can be easily satisfied, but it seems significantly more difficult to simultaneously satisfy both requirements. However, putting together our examples for predicates that satisfy each requirement on its own, we observe that for any injective one-way function f mapping n-bit inputs to $\ell(n)$-bit outputs, it holds that $\mathcal{P}(x,r) = \langle f(x), r\rangle$ is a correlated-input $(\ell(n), \delta(n))$-balanced one-way predicate, where $\delta(n) = 0$.[5] This shows

[5]This follows from our above observation that $\mathcal{P}(x,r) = \langle f(x), r\rangle$ satisfies the first requirement for any injective f, and satisfies the second requirement for any one-way f.

that the existence of such a predicate is unconditionally essential for the existence of an injective one-way function.

In addition, in Appendix A we show that the existence of a correlated-input balanced one-way predicate is also implied by that of various primitives that are secure against related-key attacks. These include, for example, related-key pseudorandom functions (e.g., [BK03, Luc04, BC10, LMR14, AW14]) and related-seed pseudorandom generators (e.g., [GL10]). Unlike injective one-way functions, these primitives seem rather unstructured, yet still suffice for constructing correlated-input balanced one-way predicates.

Our injective one-way function. Given any correlated-input $(\ell, 1/4)$-balanced one-way predicate $\mathcal{P}$, we present a construction of an injective one-way function by relying on a hitting-set generator H that fools deterministic circuits whose size is roughly that of $\mathcal{P}$'s given implementation. Our construction applies to any function $\ell = \ell(n)$ of the security parameter $n \in \mathbb{N}$ (recall that $\ell(n)$ denotes the length of $\mathcal{P}$'s second input r), as long as it is upper bounded by some polynomial (e.g., $\ell(n) = \log^2(n)$, $\ell(n) = n^2$). In what follows we first describe the construction assuming that $\ell(n) = O(\log n)$, as this case already sheds initial light on some of the main ideas underlying the construction. In fact, assuming that $\ell(n) = O(\log n)$ the construction is fully black box, and the hitting-set generator is not needed. Then, we show that the construction extends to any polynomial $\ell(n)$ by relying on a hitting-set generator.

Let $\mathcal{P}$ be a correlated-input $(\ell, 1/4)$-balanced one-way predicate where $\ell(n) = O(\log n)$, and denote by $r_{n,1}, \ldots, r_{n,L(n)}$ all $L(n) = 2^{\ell(n)}$ possible $\ell(n)$-bit strings for any $n \in \mathbb{N}$ (note that $L = L(n)$ is polynomial given that $\ell(n) = O(\log n)$). Then, we claim that the function

$$g(x) = \left(\mathcal{P}(x, r_{|x|,1}), \ldots, \mathcal{P}(x, r_{|x|,L(|x|)})\right)$$

is both injective and one way:

- The injectivity of g follows from the fact that $\mathcal{P}$ is balanced: For any distinct $x, x' \in \{0,1\}^n$, as long as $\Pr[\mathcal{P}(x,r) = \mathcal{P}(x',r)] < 1$, where the probability is taken over the choice of a uniform $r \in \{0,1\}^{\ell(n)}$, this means that there exists at least one value $r \in \{0,1\}^{\ell(n)}$ for which $\mathcal{P}(x,r) \neq \mathcal{P}(x',r)$, and therefore $g(x) \neq g(x')$.
- The one-wayness of g follows from the fact that $\mathcal{P}$ is one-way for correlated inputs: For any sequence of values $r_1, \ldots, r_T$ the function mapping x to the sequence of values $\mathcal{P}(x, r_1), \ldots,$ $\mathcal{P}(x, r_T)$ is a one-way function of x. This holds, in particular, for the sequence of values $r_{n,1}, \ldots, r_{n,L(n)}$, and thus g is a one-way function.

Now suppose that $\mathcal{P}$ is a correlated-input $(\ell, 1/4)$-balanced one-way predicate where $\ell(n)$ may be any polynomial. Here, we can no longer define g as above by enumerating over all possible $\ell(n)$-bit strings to be used as $\mathcal{P}$'s second input r. All we need, however, is to enumerate over a carefully-chosen set $r_1, \ldots, r_T$ such that for any distinct $x, x' \in \{0,1\}^n$ there exists a value $r \in \{r_1, \ldots r_T\}$ such that $\mathcal{P}(x,r) \neq \mathcal{P}(x',r)$. This is exactly the type of guarantee that is provided by a hitting-set generator, and enables us to argue that the following function g is both injective and one way: On input $x \in \{0,1\}^n$ our function $g : \{0,1\}^* \rightarrow \{0,1\}^*$ first uses a hitting-set generator H that fools circuits whose size is roughly the size of $\mathcal{P}$'s implementation for obtaining a sequence of values $r_1, \ldots, r_{T(n)} \in \{0,1\}^{\ell(n)}$, and then outputs the value

$$g(x) = \left(\mathcal{P}(x, r_1), \ldots, \mathcal{P}(x, r_{T(n)})\right).$$

In Section 3 we prove that the injectivity of g follows from the fact that $\mathcal{P}$ is balanced and H is a hitting-set generator, whereas the one-wayness of g follows from the fact that $\mathcal{P}$ is one way for correlated inputs as above. Moreover, we show that by augmenting $\mathcal{P}$ with a trapdoor argument, our construction generalizes to an injective trapdoor function. We refer the reader to Section 3 for the formal details.

Applying Rudich's impossibility to correlated-input predicates. We now briefly overview Rudich's approach while pointing out the adjustments required in order to apply it to correlated-input balanced one-way predicates. Let $\mathcal{O} = \{\mathcal{O}_n\}_{n\in\mathbb{N}}$ be an oracle, where each $\mathcal{O}_n$ is uniformly chosen from some function family $\mathfrak{S}_n$, and let C be an oracle-aided circuit guaranteeing that $C^{\mathcal{O}}$ implements an injective function for any $\mathcal{O} \in \{\mathfrak{S}_n\}_{n\in\mathbb{N}}$. In the case of Rudich's proof, $\mathfrak{S}_n$ is the family of all functions mapping n bits to n bits, and hence $\mathcal{O}$ is simply a random length-preserving function. In our case, $\mathfrak{S}_n$ is the set of all $(\ell(n) = n, \delta(n) = 2^{-n/3})$-balanced predicates; i.e., predicates taking inputs in $\{0,1\}^n \times \{0,1\}^n$, such that for every distinct $x, x' \in \{0,1\}^n$ it holds that $\left|\Pr_{r\leftarrow\{0,1\}^n}\left[\mathcal{O}_n(x,r) = \mathcal{O}_n(x',r)\right] - 1/2\right| \leq 2^{-n/3}$. We set $\ell(n) = n$ for the sake of simplicity, but the proof holds for any super-logarithmic ℓ with minor adjustments.

Rudich's proof then considers an adversary that makes a polynomial number of queries to $\mathcal{O}$ and always succeeds in inverting $C^{\mathcal{O}}(x^*)$ for any input x^*. On input $y^* = C^{\mathcal{O}}(x^*)$, the adversary A proceeds in iterations, where in each iteration it arbitrarily picks a value x and a possible oracle $\mathcal{O}'$ that is consistent with what it has learned so far on $\mathcal{O}$, such that $y^* = C^{\mathcal{O}'}(x)$. A then checks if $C^{\mathcal{O}}(x) = y^*$ (if so $x = x^*$), and if not, queries $\mathcal{O}$ with all queries in the execution of $C^{\mathcal{O}'}(x)$ that were not already known. The main observation is that in each iteration, the adversary either learns a new query made in the evaluation of $C^{\mathcal{O}}(x^*)$, or finds the correct pre-image $x = x^*$ of y^*. Hence, if C makes at most q oracle queries, then A is guaranteed to find x^* within $q+1$ iterations.

In order to prove this main observation, suppose that in some iteration A does not learn a new query made in the evaluation of $C^{\mathcal{O}}(x^*)$ nor does it hold that $x = x^*$. This means that from A's point of view, the oracles have so far been defined on disjoint sets of inputs. Now, the idea is that $\mathcal{O}$ and $\mathcal{O}'$ can be "glued" together to form a third oracle $\mathcal{O}'' \in \mathfrak{S}$ such that $C^{\mathcal{O}''}(x) = C^{\mathcal{O}''}(x^*) = y^*$, contradicting the injectivity guarantee of C. In the case of Rudich's proof, this is straightforward: since $\mathfrak{S}$ is the family of *all* length-preserving functions, $\mathcal{O}''$ can simply be any oracle that is consistent with the answers of $\mathcal{O}$ and $\mathcal{O}'$ to the queries made during the evaluations of $C^{\mathcal{O}}(x^*)$ and $C^{\mathcal{O}'}(x)$, respectively, and can be arbitrarily defined everywhere else. In our case, we need to show that we can complete $\mathcal{O}''$ to be balanced for every input length.

More generally, this shows that Rudich's proof does not only apply to length-preserving functions or to correlated-input balanced predicates, but in fact to any function family $\mathfrak{S}$ that is "sufficiently unstructured" in order to guarantee the following property: For any two functions $\mathcal{O}, \mathcal{O}' \in \mathfrak{S}$ and for any two *disjoint* sets of "not too short" inputs X and X' *of polynomial size*, there exists a function $\mathcal{O}'' \in \mathfrak{S}$ that agrees with $\mathcal{O}$ on the set X and agrees with $\mathcal{O}'$ on the set X'. We have provided two examples for such families: All length-preserving functions (i.e., where $\mathcal{O}$ a random oracle) and all balanced predicates. Of course, not all families exhibit this property as some primitives — and in particular injective one-way functions — do imply injective one-way functions in a black-box manner. For example, if we consider $\mathfrak{S} = \{\mathfrak{S}_n\}_{n\in\mathbb{N}}$ where $\mathfrak{S}_n$ is the set of all permutations on n-bit strings, then this is obviously not the case even for X and X' of size one. For any $n \in \mathbb{N}$ and any distinct $x, x' \in \{0,1\}^n$, if $\mathcal{O}(x) = \mathcal{O}'(x')$, then no function $\mathcal{O}'' \in \mathfrak{S}$ can agree both with $\mathcal{O}$ on input x and with $\mathcal{O}'$ on input x', as this will contradict the injectivity of $\mathcal{O}''$.

Two final remarks are in order. First, one still needs to show that our balanced predicate oracle is hard to invert for correlated inputs. Roughly speaking, this follows from the fact that a truly uniform predicate is correlated-input one way, and is also balanced with an overwhelming probability. Second, our proof readily extends to rule out black-box constructions of the seemingly weaker *partially-*

injective one-way functions from our strengthened variant of $\mathcal{P}$ that is augmented with a trapdoor algorithm. We refer the reader to Section 4 for the formal details.

1.3 Related Work

The power of black-box vs. non-black-box constructions. Our work shows a gap between the power of black-box constructions and the power of non-black-box constructions both in the private-key setting and in the public-key setting.

Such a gap in the private-key setting was previously identified by Mahmoody and Pass [MP12] who proved that one-way functions cannot be used in a black-box manner for constructing a non-interactive commitment scheme. Combining their negative result with the above-mentioned positive result of Barak et al. implies that, under a standard worst-case hardness assumption, the existence of a one-way function is equivalent to that of a non-interactive commitment scheme, although it is strictly weaker when restricted to black-box constructions.[6]

Our work identifies such a gap in the public-key setting as well, by showing that augmenting our primitive $\mathcal{P}$ with a trapdoor algorithm enables a non-black-box construction of a trapdoor function (while Rudich's impossibility result still holds), whereas the construction of Barak et al. does not seem to enable an extension to the public-key setting. An additional such gap in the public-key setting was identified by Döttling and Garg [DG17] who presented a breakthrough construction of an identity-based encryption scheme based on the computational Diffie-Hellman assumption, circumventing the impossibility result of Papakonstantinou et al. [PRV12] in the generic-group model.

Derandomization in cryptography. When compared to the work of Barak, Ong and Vadhan [BOV07] and other applications of derandomization in similar scenarios (e.g., [Lau83, Nao91, DN07, DNR04, BV17]), our work exhibits the following main differences.

- The underlying cryptographic building block and the resulting primitive in our work are incomparable to those in their work: We rely on a seemingly stronger cryptographic building block (specifically, a correlated-input balanced one-way predicate in our work vs. a one-way function in their work), and obtain a seemingly stronger primitive (an injective one-way function in our work vs. a partially-injective one-way function in their work). A natural question that arises in this context is whether or not our two approaches can be combined and yield a non-black-box construction of an injective one-way function based on any one-way function.

- We rely on the existence of a hitting-set generator that fools *deterministic circuits*, whereas Barak et al. rely on the seemingly incomparable assumption that there exists a hitting-set generator that fools *co-non-deterministic algorithms*. In turn, our transformation relies on the assumption that $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ has a function of deterministic circuit complexity $2^{\Omega(n)}$, whereas Barak et al. rely on the assumption that $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ has a function that is not computable for infinitely many input lengths by a probabilistic non-deterministic algorithm that runs in sub-exponential time.

- Following the work of Barak et al. derandomization using pseudorandom generators was also applied in the recent work of Bitansky and Vaikuntanathan [BV17] (both motivated by the classic applications of derandomization techniques in similar settings [Lau83, Nao91, DN07, DNR04]). The common theme underlying these applications is to derandomize an "almost perfectly correct" primitive into a "perfectly correct" one. This seems somewhat incomparable

[6]As pointed out by Mahmoody and Pass [MP12], this is different from the results of Barak [Bar01] and Goldreich and Krawczyk [GK96] which provide separations between the power of black-box and non-black-box *proofs of security*.

to our work, where our starting point is not an "almost perfectly correct" injective one-way function, but rather our new notion of a correlated-input balanced one-way predicate.

Indeed, it would seem that using an "almost perfectly correct" injective one-way function as our starting point is not enough. Consider for example a collection of functions, where all of them are one way, and most of them are injective. A standard attempt to apply derandomization techniques to construct an injective one-way function from such a collection may naturally rely on the following idea: Given an input x, use a hitting-set generator to choose a small subset of the functions in the collection, evaluate all of these functions on the same input x, and concatenate their outputs. The properties of the hitting-set generator indeed guarantee that the resulting function is injective (since at least one of the functions chosen by the generator is injective), but unfortunately there is no guarantee that this function is actually one way. A similar problem will arise when trying to start with a single function that is almost injective in the sense that it has only a few collisions. Our new primitive $\mathcal{P}$ is just strong enough to enable the construction of an injective one-way function by applying such techniques, yet still weak enough so that Rudich's black-box separation directly applies to it.

Strengthening the framework of black-box constructions. In recent years there have been several approaches for extending the framework of black-box impossibility results to capture various non-black-box techniques. For example, Brakerski et al. [BKS+11] and Asharov and Segev [AS15] showed that various non-black-box constructions that are based on non-interactive zero-knowledge proofs and indistinguishability obfuscation [BGI+12, GGH+13], respectively, can in fact be modeled in a black-box manner. This enabled them to prove various limitations on the power of these two primitives even when used in a particular non-black-box manner. Subsequently, Garg et al. [GMM17] refined the framework of Asharov and Segev to also account for "self-calls" of some primitives that might receive circuits as input (e.g., indistinguishability obfuscation).

Baecher, Brzuska and Fischlin [BBF13] considered more fine-grained variants of black-box constructions. Among their definitions, they considered constructions where the correctness or security guarantees need hold only for the case when the underlying primitive or the adversary in the security reduction are assumed to be efficient. They also went a step further, to consider a more subtle definition in which the security reduction may depend on some parameters of the assumed adversary (such as running time, success probability, etc.), even though its *access* to the adversary may still be black box. These notions seem related to, but do not precisely capture our non-black-box construction of an injective one-way function, which makes use of knowledge of the implementation size *of the underlying primitive* (with a security proof that makes black-box use of the adversary).

Most relevant to our work is the work of Pass, Tseng and Venkitasubramaniam [PTV11] that rules out constructions of various cryptographic primitives (e.g., one-way permutations, collision-resistant hash functions, constant-round statistically-hiding commitments, and constant-round black-box zero-knowledge proofs for NP) based on one-way functions, where the implementation of the underlying one-way function can be used in an arbitrary manner both within the construction and within the security proof, but the adversary may only be used in a black-box manner within the proof of security[7]. Their results are based on average-case strengthenings of the traditional assumption that coNP is not contained in AM. As Pass et al. pointed out, their approach does not seem to extend to ruling out constructions of injective one-way functions (as such functions may not be size-verifiable in general).

[7]This is exactly our case: We need a bound on the size of the underlying primitive's implementation both for the construction and for security proof, but the adversary is used in a black-box manner.

More recently, the work of Dachman-Soled [Dac16] strengthened the black-box barrier of constructing a public-key encryption scheme based on one-way functions [IR89] by relying on somewhat similar assumptions. Roughly speaking, her work considers non-adaptive constructions, where both the underlying one-way function and the adversary are used in a black-box manner by the construction and the security proof, respectively, but the security proof is allowed to rely on the implementation of the underlying one-way function in an arbitrary manner (this class of constructions seems orthogonal to our construction).

1.4 Open Problems

Circumventing other black-box barriers. A natural question that arises is whether we can rely on worst-case assumptions and similar techniques to those we use in order to circumvent other known and long-standing black-box impossibility results. In particular, can such techniques be useful in obtaining a key-agreement protocol from any one-way function or from slightly stronger yet symmetric-key primitives, or in constructing collision-resistant hash functions from any one-way function; circumventing the black-box separation results of Impagliazzo and Rudich [IR89] and of Simon [Sim98], respectively? Conversely, can one enhance the aforementioned impossibility results in a way that will provide evidence that such constructions are unlikely to exist? We refer the reader to Section 1.3 for a discussion on recent approaches to broaden the black-box separations framework.

Correlated-input balanced one-way predicates vs. one-way functions. Our new primitive $\mathcal{P}$ seems to be somewhat stronger than "plain" one-way functions, yet at least from a structural point of view, the added requirement is fairly modest and it seems much weaker than the injectivity requirement of injective one-way functions. A central open question is then the following: Can one construct a correlated-input balanced one-way predicate from any one-way function, resulting – when combined with our result (and a worst-case complexity assumption) – in a construction of an injective one-way function from any one-way function? Alternatively, can it be shown that such a construction is impossible in a black-box manner, thus establishing that a black-box barrier between general one-way functions and their injective counterparts still exists?

Per the latter possibility, it seems that the structural properties of $\mathcal{P}$ are weak enough, so that at least the techniques underlying Rudich's approach cannot be applied to ruling out black-box constructions of $\mathcal{P}$ from one-way functions. More broadly, any separation that aims to derive a contradiction to $\mathcal{P}$'s balance requirement (the first property in Definition 1.1) will have to fundamentally deviate from Rudich's technique due to the following observation. Suppose C is a candidate implementation of an (ℓ, δ)-balanced predicate with respect to some oracle $\mathcal{O}$, and say we partially fix $\mathcal{O}$ so that the output of $C^{\mathcal{O}}$ is determined for a subset X of its possible inputs of length $n + \ell(n)$. Even if X is of exponential size (in n), then $C^{\mathcal{O}}$ might still be (ℓ, δ)-balanced for a non-negligible δ, which is enough for our needs of constructing an injective one-way function.

Constructing correlated-input balanced trapdoor predicates. In the current state of affairs, candidates for injective trapdoor functions are scarce. Most candidates rely on specific number-theoretic or lattice based assumptions, and general constructions from other cryptographic primitives either rely on very strong assumptions such as sub-exponential indistinguishability obfuscation [BPW16] or are proven in the random oracle model [BHS$^+$98]. We thus view the construction of our trapdoor version of $\mathcal{P}$ from new assumptions as a very interesting open problem, as this will imply new constructions for injective trapdoor functions. More specifically, can the trapdoor version of $\mathcal{P}$ be obtained from public-key encryption (perhaps with additional symmetric primitives)? Can enhancing the latter's security properties help in such a transformation (similarly to the symmetric case, in which we were able to trade strong security guarantees of related-key secure pseudorandom functions for the structural ones of $\mathcal{P}$)?

Weakening the derandomization-related assumption. Our construction of an injective one-way function is based on the existence of a hitting-set generator, which in turn is known to exist under the assumption of a non-uniform circuit lower bound (namely, that $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ has a function of deterministic circuit complexity $2^{\Omega(n)}$). Can this assumption be weakened? More specifically, can similar results be obtained using weaker types of hitting-set generators or pseudorandom generators, known to exist under seemingly weaker complexity assumptions? For example, can results of similar nature be based on the seemingly weaker assumption that $\mathsf{P} = \mathsf{BPP}$, which Goldreich [Gol11] showed to yield certain uniform versions of pseudorandom generators?

Implications to extensions of Rudich's work. A variety of extensions have been developed to Rudich's impossibility result, including for example [BKS$^+$11, MM11, AS15, AS16b, BDV17, RSS17]. Our result does not directly imply that all of these extensions may be circumvented as well, since they deal with primitives that seem either significantly stronger than injective one-way functions (e.g., public-key primitives [BKS$^+$11, AS15] and specific forms of injective one-way functions [MM11, AS16b]), or incomparable to injective one-way functions (e.g., bounded-TFNP instances [RSS17]), and are currently not known to be implied by our notion of a correlated-input balanced one-way predicate. An interesting problem that arises given these extensions is to extend our approach to such stronger or incomparable primitives.

1.5 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we introduce our notation as well as the basic cryptographic primitives that we consider in this paper. In Section 3 we present our constructions of an injective one-way function and of an injective trapdoor function. In Section 4 we show that Rudich's impossibility result applies not only to constructions based on one-way functions, but also to constructions based on correlated-input balanced one-way predicates (and even when augmented with a trapdoor algorithm). Finally, in Appendix A we show that the existence of such predicates is unconditionally implied not only by that of any injective one-way function, but also by that of various cryptographic primitives that are secure against related-key attacks.

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. The *statistical distance* between two distributions X and Y over a finite domain Ω is $\mathsf{SD}(X, Y) = \frac{1}{2}\sum_{\omega\in\Omega} |X(\omega) - Y(\omega)|$. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. A function $\nu : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for any polynomial $p(\cdot)$ there exists an integer N such that for all $n > N$ it holds that $\nu(n) \leq 1/p(n)$.

2.1 Hitting-Set Generators

We rely on the following standard notion of a hitting-set generator, as formalized by Goldreich et al. [GVW11], for the class of deterministic circuits (see also [Sip88, CG89, And94, ACR$^+$97, LLS$^+$97, ACR98, GVW11] and the references therein).

Definition 2.1. A deterministic polynomial-time algorithm H is a *hitting-set generator that fools deterministic circuits* if for every $n, t \in \mathbb{N}$ the generator H on input $\left(1^n, 1^t\right)$ outputs a set $\mathcal{S}$ such that the following hold:

- $\mathcal{S} \subseteq \{0,1\}^n$.
- For every circuit $C : \{0,1\}^n \to \{0,1\}$ of size at most t for which

$$\Pr_{x \leftarrow \{0,1\}^n}[C(x) = 1] \geq 1/4,$$

  there exists some $x^* \in \mathcal{S}$ such that $C(x^*) = 1$.

Any *pseudorandom* generator [NW94] that fools deterministic circuits and has a logarithmic seed length immediately gives rise to such a hitting-set generator (by having H enumerate over all possible seeds). This implies the following corollary on which we rely for our constructions in Section 3:

Corollary 2.2 ([NW94, IW97]). *If there exists a function $f \in DTIME(2^{O(n)})$ with deterministic circuit complexity $2^{\Omega(n)}$, then there exists a hitting-set generator that fools deterministic circuits.*

2.2 Injective and Partially-Injective One-Way Functions

In this paper we rely on the following standard notions of one-way functions and injective one-way functions (see, for example, [Gol01]), as well as on the notion of partially-injective one-way functions due to Barak, Ong and Vadhan [BOV07].

Definition 2.3. An efficiently-computable function $f : \{0,1\}^* \to \{0,1\}^*$ is *one way* if for every probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\cdot)$ such that

$$\Pr_{x \leftarrow \{0,1\}^n}\left[A(1^n, f(x)) \in f^{-1}\left(f(x)\right)\right] \leq \nu(n)$$

for all sufficiently large $n \in \mathbb{N}$.

An injective one-way function is a function that is both injective and one way. Barak, Ong, and Vadhan [BOV07] introduced the following notion of a partially-injective one-way function.

Definition 2.4 ([BOV07]). Let $m = m(n)$ be a function of the security parameter $n \in \mathbb{N}$. An efficiently-computable function $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is a *partially-injective one-way function* if it satisfies the following two requiremets:

1. For every $n \in \mathbb{N}$, every $x, x' \in \{0,1\}^n$ such that $x \neq x'$, and every $y, y' \in \{0,1\}^{m(n)}$, it holds that $f(x, y) \neq f(x', y')$ (i.e., f is injective with respect to its first input).
2. For every probabilistic polynomial-time algorithm A there exits a negligible function $\nu(\cdot)$ such that

$$\Pr_{(x,y) \leftarrow \{0,1\}^{n+m(n)}}\left[A\left(f(x, y)\right) = x\right] \leq \nu(n)$$

   for all sufficiently large $n \in \mathbb{N}$.

Note that a partially-injective one-way function with $m(n) = 0$ is in fact an injective one-way function, but for general $m(n)$ this notion seems potentially weaker than that of an injective one-way function. Barak et al. observed that any perfectly-binding non-interactive commitment scheme yields a partially-injective one-way function. Since Barak et al. derandomized Naor's commitment scheme [Nao91] into a perfectly-binding non-interactive one assuming the existence of a hitting-set generator that fools co-non-deterministic algorithms (recall Section 2.1), the following corollary follows.

Corollary 2.5 ([BOV07]). *Assuming the existence of a hitting-set generator that fools co-non-deterministic algorithms, then one-way functions imply partially-injective one-way functions.*

2.3 Injective Trapdoor Functions

We also rely in this paper on the following standard notion of a collection of trapdoor functions (see, for example, [Gol01]).

Definition 2.6. Let $m = m(n)$ be a function of the security parameter $n \in \mathbb{N}$. *A collection of trapdoor functions* is a triplet of efficient algorithms $\mathcal{F} = (G, F, F^{-1})$ satisfying the following requirements:

1. G is a probabilistic algorithm that on input 1^n, samples and outputs a public key $pk \in \{0,1\}^n$ and a corresponding trapdoor $td \in \{0,1\}^n$.[8]

2. F is a deterministic algorithm that receives as input a public key $pk \in \{0,1\}^n$ and an additional input value $x \in \{0,1\}^n$ and outputs a value $y \in \{0,1\}^{m(n)}$. We require that for every probabilistic polynomial-time algorithm A there exists a negligible function ν such that

$$\Pr_{\substack{(td,pk)\leftarrow G(1^n)\\ x\leftarrow\{0,1\}^n}}\left[F(pk, A(1^n, pk, F(pk,x))) = F(pk,x)\right] \leq \nu(n)$$

   for all sufficiently large $n \in \mathbb{N}$.

3. F^{-1} is a deterministic algorithm that on input $(td, F(pk,x))$ has the following guarantee: For any $n \in \mathbb{N}$, (td, pk) in the range of $G(1^n)$ and $x \in \{0,1\}^n$, it holds that $F^{-1}(td, F(pk,x))$ outputs $x' \in \{0,1\}^n$ such that $F(pk,x') = F(pk,x)$.

We say that $\mathcal{F}$ is a collection of injective trapdoor functions if for every $n \in \mathbb{N}$ and any (td, pk) in the range of $G(1^n)$ the function $F(pk, \cdot)$ is injective.

3 Our Constructions

In this section we present our non-black-box constructions of an injective one-way function (see Section 3.1) and an injective trapdoor function (see Section 3.2).

3.1 An Injective One-Way Function

In this section we present our non-black-box construction of an injective one-way function from any correlated-input balanced one-way predicate and any hitting-set generator that fools deterministic circuits. More formally, our construction relies on the following two building blocks:

- A correlated-input $(\ell, 1/4)$-balanced one-way predicate $\mathcal{P}$ (recall Definition 1.1), where $\ell(n)$ may be upper bounded by any fixed polynomial (e.g., $\ell(n) = \log^2(n)$, $\ell(n) = n^2$). Let $t = t(n)$ be an upper bound on the size of the circuit computing $\mathcal{P}(x,r)$ for inputs $x \in \{0,1\}^n$ (recall that $r \in \{0,1\}^{\ell(n)}$).

- A hitting-set generator H that fools deterministic circuits. Denote by $T = T(n)$ the size of the set $\mathcal{S}$ that is produced by the generator H on input $(1^{\ell(n)}, 1^{2t(n)+c})$ for a constant c to be determined later. As discussed in Section 2.1, such a generator exists based on the worst-case assumption that $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ has a function with deterministic circuit complexity $2^{\Omega(n)}$.

[8]Definition 2.6 assumes that the lengths of the public key and of the trapdoor are equal to the security parameter n. This is for simplicity only, and in both cases one may replace n with any length that is polynomial in n.

We note that the choice of the constant 1/4 that parameterizes both of our building blocks is rather arbitrary. More generally, the construction may rely on any (ℓ, δ)-balanced predicate and on any ϵ-hitting-set generator as long as $\delta + \epsilon \geq 1/2$ (in Definition 2.1 we fixed ϵ to be 1/4, but the definition readily extends to any $\epsilon \in [0,1]$).

The construction. On input $x \in \{0,1\}^n$ our function $g : \{0,1\}^* \to \{0,1\}^*$ first computes

$$H\left(1^{\ell(n)}, 1^{2t(n)+c}\right) = \left(r_1, \ldots, r_{T(n)}\right) \in \{0,1\}^{\ell(n) \times T(n)},$$

where $c > 0$ is a fixed constant that we determine later on, and then outputs the value

$$g(x) = \left(\mathcal{P}(x, r_1), \ldots, \mathcal{P}(x, r_{T(n)})\right) \in \{0,1\}^{T(n)}.$$

The following theorem states that g is an injective one-way function based on our assumptions on the underlying building blocks $\mathcal{P}$ and H:

Theorem 3.1. *Assuming that $\mathcal{P}$ is a correlated-input $(\ell, 1/4)$-balanced one-way predicate and that H is a hitting-set generator that fools deterministic circuits, the function g is an injective one-way function.*

Proof. We first prove that g is injective (Claim 3.2) by relying on the fact that $\mathcal{P}$ is balanced for correlated inputs (recall the first requirement of Definition 1.1) and on the hitting property of H. Then, we prove that g is one-way (Claim 3.3) by relying on the fact that $\mathcal{P}$ is one-way for correlated inputs (recall the second requirement of Definition 1.1).

Claim 3.2. For every $n \in \mathbb{N}$ and for every distinct $x, x' \in \{0,1\}^n$ it holds that $g(x) \neq g(x')$.

Proof of Claim 3.2. Fix any $n \in \mathbb{N}$ and distinct inputs $x, x' \in \{0,1\}^n$, and we show that $g(x) \neq g(x')$. Consider the deterministic circuit $C = C_{x,x'}$ that on input $r \in \{0,1\}^{\ell(n)}$ outputs 1 if and only if $\mathcal{P}(x, r) \neq \mathcal{P}(x', r)$. This circuit preforms two evaluations of $\mathcal{P}$ and one bit comparison, and thus the size of the circuit C is at most $2t(n) + c$, where c is the number of gates needed to compare two bits. The balancing property of $\mathcal{P}$ guarantees that

$$\Pr_{r \leftarrow \{0,1\}^{\ell(n)}}\left[C(r) = 1\right] = \Pr_{r \leftarrow \{0,1\}^{\ell(n)}}\left[\mathcal{P}(x, r) \neq \mathcal{P}(x', r)\right] \geq 1/4.$$

Now, the hitting-set generator H guarantees that for the sequence of values

$$H\left(1^{\ell(n)}, 1^{2t(n)+c}\right) = \left(r_1, \ldots, r_{T(n)}\right) \in \{0,1\}^{\ell(n) \times T(n)}$$

there exists some $i \in [T(n)]$ such that $C(r_i) = 1$, which is equivalent to $\mathcal{P}(x, r_i) \neq \mathcal{P}(x', r_i)$, and thus $g(x) \neq g(x')$. ∎

Claim 3.3. The function g is one way.

Proof of Claim 3.3. Assume towards a contradiction that there exists a probabilistic polynomial-time algorithm A and a polynomial $p(\cdot)$ such that

$$\Pr_{x \leftarrow \{0,1\}^n}\left[A(1^n, g(x)) = x\right] \geq 1/p(n)$$

for infinitely many values of $n \in \mathbb{N}$.

We construct a probabilistic polynomial-time algorithm A' that contradicts the correlated-input one-wayness of $\mathcal{P}$. On input 1^n the algorithm A' participates in the experiment $\mathsf{Invert}_{\mathcal{P},A'}(n)$ as follows:

1. The algorithm A' outputs the sequence of correlated inputs $\left(r_1, \ldots r_{T(n)}\right) = H\left(1^{\ell(n)}, 1^{2t(n)+c}\right)$, and then receives $\mathcal{P}\left(x, r_1\right), \ldots, \mathcal{P}\left(x, r_{T(n)}\right)$ for a uniformly sampled $x \leftarrow \{0,1\}^n$.
2. The algorithm A' invokes A on the input $\left(1^n, \left(\mathcal{P}\left(x, r_1\right), \ldots, \mathcal{P}\left(x, r_{T(n)}\right)\right)\right)$, and returns its output.

Since $g(x) = \mathcal{P}\left(x, r_1\right), \ldots, \mathcal{P}\left(x, r_{T(n)}\right)$, it holds that

$$\Pr\left[\mathsf{Invert}_{\mathcal{P},A'}(n) = 1\right] = \Pr_{x \leftarrow \{0,1\}^n}\left[A(1^n, g(x)) = x\right] \geq 1/p(n)$$

for infinitely many values of $n \in \mathbb{N}$, which contradicts the correlated-input one-wayness of $\mathcal{P}$. Therefore, the function g is one way. ■

This concludes the proof of Theorem 3.1. ■

3.2 An Injective Trapdoor Function

We now turn to extend our approach to injective *trapdoor* functions. Loosely speaking, we augment our primitive $\mathcal{P}$ with a trapdoor algorithm $\mathcal{P}^{-1}$, and show that an extension of the construction presented in Section 3.1 yields an injective trapdoor function. Informally, knowledge of a trapdoor enables $\mathcal{P}^{-1}$ to find an $x \in \{0,1\}^n$ such that $\mathcal{P}(x,r) = b$ for each pair $(r,b) \in \{0,1\}^{\ell(n)} \times \{0,1\}$ in a set S of such pairs that is given as input to the algorithm, with the proviso that S provides "sufficient information" about x. This last condition may be formalized as a boolean set function $\phi : \left(\{0,1\}^{\ell}\right)^* \to \{0,1\}$ with the interpretation that a set is mapped to 1 if and only if it is "sufficiently rich". Informally, a reasonable choice of a function ϕ should meet two criteria:

1. For every $n \in \mathbb{N}$, it should be possible to efficiently come up with a set that satisfy ϕ. Otherwise, $\mathcal{P}^{-1}$ seems of little use.

2. ϕ should be monotone; i.e., if $S \subseteq T$ and $\phi(S) = 1$, then $\phi(T) = 1$. Intuitively, if $\phi(S) = 1$ has the interpretation that S generates "enough information" on x, then surely this is also the case for T.

A natural choice for ϕ, which we will adopt in our definition below, is a function that checks whether or not the input set contains a basis for $\mathbb{F}_2^{\ell(n)}$ (when each element $r \in \{0,1\}^{\ell(n)}$ is viewed a vector in $\mathbb{F}_2^{\ell(n)}$); that is, $\phi(S) = 1$ if and only if S contains a subset of $\ell(n)$ linearly independent r's. This choice, other than satisfying the aforementioned criteria, enables us to construct a correlated-input balanced *trapdoor* predicate (as will be defined shortly in Definition 3.4) from any injective trapdoor function, making our trapdoor predicate with respect to that choice of ϕ essential for the existence of injective trapdoor functions.

It should be noted, however, that any choice of ϕ that satisfies the above two criteria yields a predicate that can be used in a non-black-box manner to construct an injective trapdoor function via our transformation, yet (a strengthened version of) Rudich's proof shows that this is not the case when restricting ourselves to black-box constructions. Indeed, in Section 4 we show that this augmented variant of $\mathcal{P}$ cannot be used in a black-box manner to construct even a partially-injective one-way function.

The following definition naturally extends Definition 1.1 by considering a family of predicates equipped with a trapdoor algorithm, as discussed above:

Definition 3.4. Let $\ell = \ell(n)$ and let $\delta = \delta(n)$ be functions of the security parameter. A *correlated-input (ℓ, δ)-balanced trapdoor predicate* is a triplet $\mathcal{T} = (G, P, P^{-1})$ of efficiently-computable algorithms such that:

- The algorithm G on input 1^n outputs a pair $(pk, td) \in \{0,1\}^*$.
- For every $n \in \mathbb{N}$ and for every $pk \in \{0,1\}^*$ produced by $G(1^n)$, the function $P(pk, \cdot, \cdot) : \{0,1\}^n \times \{0,1\}^{\ell(n)} \to \{0,1\}$ is an (ℓ, δ)-balanced predicate. That is, for any $x, x' \in \{0,1\}^n$ such that $x \neq x'$ it holds that

$$\left| \Pr_{r \leftarrow \{0,1\}^{\ell(n)}} \left[\mathcal{P}(pk, x, r) = \mathcal{P}(pk, x', r) \right] - \frac{1}{2} \right| \leq \delta(n).$$

- For every $n, T \in \mathbb{N}$, and for every (pk, td) that is produced by $G(1^n)$, the algorithm P^{-1} satisfies the following guarantee:
  On input td and $\{(r_i, b_i)\}_{i=1}^T \in \left(\{0,1\}^{\ell(n)} \times \{0,1\}\right)^T$, if the set $\{r_i\}_{i=1}^T$ contains a subset of $\ell(n)$ linearly independent elements and there exists an $x \in \{0,1\}^n$ such that $P(pk, r_i) = b_i$ for every $i \in [T]$, then P^{-1} outputs such an x. Otherwise, P^{-1} outputs $\perp$.
- For any probabilistic polynomial-time algorithm A there exists a negligible function $\nu(\cdot)$ such that

$$\Pr\left[\mathsf{Invert}_{\mathcal{T},A}(n) = 1\right] \leq \nu(n)$$

  for all sufficiently large $n \in \mathbb{N}$, where the experiment $\mathsf{Invert}_{\mathcal{T},A}(n)$ is defined as follows:
  1. $(\mathsf{state}, r_1, \dots, r_T) \leftarrow A(1^n, pk)$ for $r_1, \dots, r_T \in \{0,1\}^{\ell(n)}$, where $T = T(n)$ may be any polynomial determined by A and $(pk, td) \leftarrow G(1^n)$.
  2. $x' \leftarrow A\left(\mathsf{state}, P_{pk}\left(x, r_1\right), \dots, P_{pk}\left(x, r_T\right)\right)$ where $x \leftarrow \{0,1\}^n$.
  3. If $x' = x$ then output 1, and otherwise output 0.

Observe that the existence of $\mathcal{T}$ is indeed essential for the existence of an injective trapdoor function. Let $\mathcal{F} = (G_F, F, F^{-1})$ be any collection of injective trapdoor functions, and consider the construction $P_{pk}(x, r) = \langle F_{pk}(x), r \rangle$. This is essentially (a keyed version of) the same construction that we had in the symmetric case, and P is thus both balanced for every pk and correlated-input one way for the same reasons as before. As for the inversion algorithm P^{-1}, note that given the construction P, every pair (r, b) in the input to P^{-1} may be interpreted as a linear equation with $\ell(n)$ variables over $\mathbb{F}_2$: $\langle F_{pk}(x), r \rangle = b$. Hence, when the input to P^{-1} contains $\ell(n)$ linearly independent r's (which is only then that it is required to return a pre-image x), it can uniquely recover $z = F_{pk}(x)$ and invoke $F_{td}^{-1}(z)$ to find x.

Similarly to Section 3.1, our construction of an injective trapdoor function is based a hitting-set-generator against deterministic circuits H, but we replace the correlated-product $(\ell, 1/4)$-balanced *one-way* predicate $\mathcal{P}$, with a correlated-product $(\ell, 1/4)$-balanced *trapdoor* predicate $\mathcal{T} = (G, P, P^{-1})$. As before, we let $t = t(n)$ be an upper bound on the size of the circuit computing $P_{pk}(x, r)$ for $x \in \{0,1\}^n$ and let $T = T(n)$ denote the size of $\mathcal{S} = \{r_1, \dots, r_T\}$ - the output set of H on input $(1^{\ell(n)}, 1^{2t(n)+c})$.

The construction. The construction extends that of an injective one-way function presented in Section 3.1. The main difference is that we need to make sure that the output of $F_{pk}(x)$ encodes "enough information" on x so that we may use P_{td}^{-1} to implement the inversion algorithm F_{td}^{-1}. To ensure that, when computing $F_{pk}(x)$, we will also invoke P_{pk} on $(x, e_1), \dots, (x, e_{\ell(n)})$, where $e_1, \dots, e_{\ell(n)}$ are the standard basis vectors, interpreted as binary strings of length $\ell(n)$. The output of $F_{pk}(x)$ will then consist of two parts: The first part $P_{pk}(x, r_1), \dots, P_{pk}(x, r_T)$ ensures injectivity (as in Section 3.1), while the second part $P_{pk}(x, e_1), \dots, P_{pk}(x, e_{\ell(n)})$ ensures efficient invertibility.

Concretely, given the aforementioned ingredients, we construct an injective trapdoor function $\mathcal{F} = (G_F, F, F^{-1})$ as follows:

- The algorithm G_F on input 1^n invokes $G(1^n)$, and outputs its output (pk, td).
- The algorithm F on input $(pk, x) \in \{0,1\}^n \times \{0,1\}^n$ computes $H(1^{\ell(n)}, 1^{2t(n)+c}) = (r_1, \ldots, r_T)$, and outputs

$$F_{pk}(x) = \left(P_{pk}(x, r_1), \ldots P_{pk}(x, r_T), P_{pk}(x, e_1), \ldots, P_{pk}(x, e_{\ell(n)})\right).$$

- The algorithm F^{-1} on input $(td, y) \in \{0,1\}^n \times \{0,1\}^{T+\ell(n)}$ computes $H(1^{\ell(n)}, 1^{2t(n)+c}) = (r_1, \ldots, r_T)$, and outputs

$$F^{-1}_{td}(y) = P^{-1}_{td}\left((r_1, y_1), \ldots, (r_T, y_T), (e_1, y_{T+1}), \ldots, (e_{\ell(n)}, y_{T+\ell(n)})\right)$$

  where y_i denotes the ith bit of y for every $i \in \{1, \ldots, T + \ell(n)\}$.

Theorem 3.5. *Assuming that $\mathcal{T} = (G, P, P^{-1})$ is a correlated-input $(\ell, 1/4)$-balanced trapdoor predicate and that H is a hitting-set generator that fools deterministic circuits, the triplet $\mathcal{F} = (G_F, F, F^{-1})$ is an injective trapdoor function.*

Proof. We first note that for every $n \in \mathbb{N}$ and for every $pk \in \{0,1\}^n$, the injectivity of $F_{pk}(\cdot)$ follows from the fact that P_{pk} is balanced and from the properties of the hitting-set-generator H (the formal proof is essentially identical to that of Claim 3.2). In addition, the one-wayness of F follows directly from the one-wayness of P for correlated inputs as defined via the experiment $\mathsf{Invert}_{\mathcal{T},A}(n)$ in Definition 3.4 (the proof is again essentially identical to that of Claim 3.3).

It is thus left to prove the correctness of the inversion algorithm F^{-1}. Indeed, by the definition of the algorithms G_F and F, for any $n \in \mathbb{N}$, $(pk, td) \leftarrow G_F(1^n)$, $x \in \{0,1\}^n$ and $y = F_{pk}(x)$, it holds that $y_i = P_{pk}(x, r_i)$ for every $i \in [T(n)]$ and $y_{T+i} = P_{pk}(x, e_i)$ for every $i \in [\ell(n)]$. Moreover, by the injectivity of $F_{pk}(\cdot)$, for any $x' \in \{0,1\}^n$ such that $x' \neq x$, there exists an index $i \in [T(n)]$ for which $P_{pk}(x, r_i) = (F_{pk}(x))_i \neq y_i$. Put in words, x is the only element in $\{0,1\}^n$ for which $P_{pk}(x, r_i) = y_i$ for every $i \in \{0,1\}^n$. Moreover, the set $\{r_i\}_{i=1}^{T} \cup \{e_i\}_{i=1}^{\ell(n)}$ on which P^{-1}_{td} is invoked on by F^{-1}_{td} indeed contains a subset of $\ell(n)$ linearly independent elements (in particular, it includes $e_1, \ldots, e_{\ell(n)}$). It follows by the guarantee of P^{-1}_{td} that indeed $F^{-1}_{td}(y) = x$. ■

4 Applying Rudich's Impossibility to Correlated-Input Predicates

In this section we show that Rudich's impossibility result [Rud88] can be applied to correlated-input balanced trapdoor predicates. That is, we show that there is no black-box construction of an injective one-way function from such a predicate. In fact, we prove a stronger result, showing that there is no black-box construction of a *partially-injective* one-way function (as defined by Barak, Ong and Vadhan [BOV07]) from such a predicate (recall Definition 2.4). Since any injective trapdoor function is also an injective (and a partially-injective) one-way function, it trivially follows that the former also cannot be constructed in a black-box manner from our predicate. We prove the following theorem:

Theorem 4.1. *There is no black-box construction of a partially-injective one-way function based on a correlated-input $(\ell(n), \delta(n))$-balanced trapdoor predicate, where $\ell(n) = n$ and $\delta(n) = 2^{-n/3}$.*

We note that, as with Rudich's original statement, the above theorem applies even to *semi*-black-box constructions (i.e., cases where the construction itself is black box, but adversaries may be used in a non-black-box manner within the proof of security – see [RTV04] for more details). In addition, we note that our choice of $\ell(n) = n$ is done purely for simplicity, and our proof applies to any super-logarithmic $\ell(n)$ (recall that a logarithmic $\ell(n)$ does imply an injective one-way function in a black-box manner – see Section 1.2).

In what follows we first describe the oracle that enables us to prove our result (essentially replacing Rudich's random function with a random predicate and complementing it with a trapdoor oracle). We describe and analyze (a slightly modified version of) Rudich's attacker with respect to this oracle, showing that it can invert any partially-injective one-way function. Then, we show that this oracle is an exponentially-secure correlated-input balanced trapdoor predicate for poly-query adversaries. Theorem 4.1 then immediately follows (see, for example, [Rud88, IR89, RTV04]). Throughout our proof we rely on the following standard notion of a q-query algorithm:

Definition 4.2. Let A be an oracle-aided algorithm and let $q = q(n)$ be a function of the security parameter $n \in \mathbb{N}$. Then, A is a *q-query algorithm* if for any $n \in \mathbb{N}$ it holds that A issues at most $q(n)$ oracle queries when invoked on inputs of length n.

The oracle. Our oracle is a triplet $\mathcal{T} = (\mathcal{G}, \mathcal{P}, \mathcal{P}^{-1}) = \{(\mathcal{G}_n, \mathcal{P}_n, \mathcal{P}_n^{-1})\}_{n\in\mathbb{N}}$ of three sub-routines. For every $n \in \mathbb{N}$, the functions $\mathcal{G}_n, \mathcal{P}_n$ and $\mathcal{P}_n^{-1}$ are defined as follows:

- The function $\mathcal{G}_n : \{0,1\}^n \to \{0,1\}^n$ is a uniformly chosen function from $\{0,1\}^n$ to $\{0,1\}^n$. Looking ahead, $\mathcal{G}_n$ will be used for mapping trapdoors to corresponding public keys.
- For any $pk \in \{0,1\}^n$ the function $\mathcal{P}_n(pk, \cdot, \cdot) : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is a predicate sampled uniformly at random from all predicates of suitable input-length that are correlated-input $\delta(n)$-balanced, independently of $\mathcal{P}_n(pk', \cdot, \cdot)$ for any $pk' \neq pk$. That is, for any $pk \in \{0,1\}^n$, the predicate $\mathcal{P}_n(pk, \cdot, \cdot)$ is sampled uniformly subject to the condition that for any distinct $x, x' \in \{0,1\}^n$ it holds that

$$\left| \Pr_{r \leftarrow \{0,1\}^n} \left[\mathcal{P}_n(pk, x, r) = \mathcal{P}_n(pk, x', r) \right] - \frac{1}{2} \right| \leq 2^{-n/3}.$$

- For any $td \in \{0,1\}^n$, the function $\mathcal{P}^{-1}(td, \cdot) : (\{0,1\}^n \times \{0,1\})^* \to \{0,1\}^n \cup \{\perp\}$ is defined as follows. For $R = \{(r_i, b_i)\}_i \in (\{0,1\}^n \times \{0,1\})^*$ define the set:

$$X_{td,R} = \{\, x \in \{0,1\}^n : \exists pk \in \{0,1\}^n \text{ s.t. } \mathcal{G}(td) = pk \wedge \forall i, \mathcal{P}(pk, x, r_i) = b_i \}\,.$$

  Then, for every $R \in (\{0,1\}^n \times \{0,1\})^*$, if $X_{td,R} \neq \emptyset$, $\mathcal{P}_n^{-1}(td, R)$ returns a uniformly chosen element in the set. Otherwise, it returns $\perp$.

We denote the set of all such oracles by $\mathfrak{S}$.

4.1 Inverting Partially-Injective One-Way Functions

Suppose F is an s-size, q-query black-box implementation of a partially-injective one-way function from the oracle $\mathcal{T}$ for some polynomially bounded $s = s(n)$ and $q = q(n)$. We assume without loss of generality that before each query of the form $(td, \{(r_i, b_i)\}_i)$ that F makes to $\mathcal{P}^{-1}$, it also obtains $pk = \mathcal{G}(td)$ via a single query to $\mathcal{G}$, and after learning $x = \mathcal{P}^{-1}(td, \{(r_i, b_i)\}_i)$ it also queries $\mathcal{P}$ with (pk, x, r_i) for each r_i (if $x = \perp$, we forgo these queries to $\mathcal{P}$). Note that as F makes at most $q(n)$

queries to $\mathcal{P}^{-1}$ and each of which involves at most $s(n)$ values of r, this adds at most $q(n) \cdot (s(n)+1)$ queries to the computation. For ease of notation we simply assume F makes the afore-described queries and continue to bound on the total number of queries made by F by $q(n)$.

The following lemma shows that for every black-box implementation F of a partially-injective one-way function from the oracle $\mathcal{T}$, there exists a poly-query adversary that on input $F^{\mathcal{T}}(x,y)$ always finds x.

Lemma 4.3. *Let $q = q(n), s = s(n)$ and let F be an s-size, q-query algorithm such that for every $\mathcal{T} \in \mathfrak{S}$ it holds that $F^{\mathcal{T}} : \{0,1\}^* \to \{0,1\}^*$ is partially injective. Then, there exists an $O(q^6 \cdot s^6)$-query algorithm A such that*

$$\Pr_{(x,y)\leftarrow\{0,1\}^{n+m(n)}}\left[A^{\mathcal{T}}\left(F^{\mathcal{T}}(x,y)\right) = x\right] = 1$$

for all sufficiently large $n \in \mathbb{N}$.

Consider the following attacker A, that on input v^* finds x^* such that there exists some y^* for which $F^{\mathcal{T}}(x^*,y^*) = v^*$:

- **Input:** A string $v^* \in \{0,1\}^*$, which is the output of $F^{\mathcal{T}}$ on input $(x^*,y^*) \in \{0,1\}^{n+m(n)}$.
- **Initialize:** A initializes a set $Q(A) = \emptyset$, to hold all query/answer pairs to $\mathcal{T}$ that A learns throughout the attack.
- **Learning:** Let $n' = 2\log(2q(n) \cdot s(n))$. A queries $\mathcal{P}$ with all queries of length at most $3n' = n' + n' + \ell(n')$, and updates $Q(A)$ accordingly.
- **Iteration:** A runs $q(n)+1$ iterations of the following three steps:
  1. Simulation: A finds a possible execution of F that is consistent with $Q(A)$ and v^*. That is, A finds inputs x, y, and an oracle $\widehat{\mathcal{T}} = (\widehat{\mathcal{G}}, \widehat{\mathcal{P}}, \widehat{\mathcal{P}}^{-1}) \in \mathfrak{S}$ that is consistent with $Q(A)$, such that $F^{\widehat{\mathcal{T}}}(x,y) = v^*$.
  2. Evaluation: A evaluates $F^{\mathcal{T}}(x,y)$ (note that the evaluation is done with the true oracle $\mathcal{T}$). In case $F^{\mathcal{T}}(x,y) = v^*$, A terminates and outputs x.
  3. Update: A queries the true oracle $\mathcal{T}$ with all queries made in the execution of $F^{\widehat{\mathcal{T}}}(x,y)$ and are not in $Q(A)$, and updates $Q(A)$ accordingly. Additionally, for any query of the form $u = (td, \{(r_i, b_i)\}_i)$ that A makes to (the true oracle) $\mathcal{P}^{-1}$ in the update phase, it also queries $\mathcal{P}$ with (pk, x, r_i) for each r_i, where x is the answer to u according to $\widehat{\mathcal{P}}^{-1}$ (if $x = \perp$, A forgoes these queries to $\mathcal{P}$), and pk is the public-key associated with td according to $\widehat{\mathcal{G}}$; i.e., $pk = \widehat{\mathcal{G}}(td)$.[9]

The success and query efficiency of A follow immediately by the following claim.

Claim 4.4. In each iteration, at least one of the following events occur:

1. A queries $\mathcal{T}$ with a query that is made by the execution of $F^{\mathcal{T}}(x^*,y^*)$, but was not in $Q(A)$ at the beginning of the iteration.
2. A finds x^* and some y for which $F^{\mathcal{T}}(x^*,y) = v^*$, and terminates.

[9]Recall that we assumed that before each query to $\mathcal{P}^{-1}$ that contains some trapdoor td, F queries $\mathcal{G}$ with td. Hence, this is also the case in the execution chosen by A in the simulation step.

Proof of Lemma 4.3 from Claim 4.4. Since $F^{\mathcal{T}}(x^*, y^*)$ makes at most $q(n)$ queries to $\mathcal{T}$, by Claim 4.4 and the pigeon-hole principle, there exists an iteration in which A finds x^* and terminates. Moreover, during the learning phase, A queries the oracle with $O\left(q(n)^6 \cdot s(n)^6\right)$ queries, and in each iteration it queries the oracle with at most $q(n) \cdot (s(n) + 2)$ new queries. Since there are at most $q(n) + 1$ iterations, A is an $O(q^6 \cdot s^6)$-query algorithm. ■

Proof of Claim 4.4. Assume towards a contradiction that in some iteration neither of the events occur. In the case event 1 does not occur, we can build a valid oracle $\widetilde{\mathcal{T}} = (\widetilde{\mathcal{G}}, \widetilde{\mathcal{P}}, \widetilde{\mathcal{P}}^{-1})$ that is consistent with $\mathcal{T}$ in the evaluation of $F^{\mathcal{T}}(x^*, y^*)$, and is consistent with $\widehat{\mathcal{T}}$ in the sampled execution $F^{\widehat{\mathcal{T}}}(x, y)$ in the following manner. We describe how to define $\widetilde{\mathcal{G}}$ and $\widetilde{\mathcal{P}}$, as this will also define $\widetilde{\mathcal{P}}^{-1}$.

For every trapdoor $td \in \{0,1\}^n$, we define $\widetilde{\mathcal{G}}(td)$ as follows:

- If td is queried during the evaluation of $F^{\mathcal{T}}(x^*, y^*)$, we let $\widetilde{\mathcal{G}}(td) = \mathcal{G}(td)$.
- If td is queried during the evaluation of $F^{\widehat{\mathcal{T}}}(x, y)$, we let $\widetilde{\mathcal{G}}(td) = \widehat{\mathcal{G}}(td)$.
- Otherwise, we set $\widetilde{\mathcal{G}}(td)$ to be some arbitrary public-key $pk \in \{0,1\}^n$ that does not appear in a query to $\mathcal{P}$ in either of the above computations (i.e., $F^{\mathcal{T}}(x^*, y^*)$ and $F^{\widehat{\mathcal{T}}}(x, y)$).

Note that since we assumed to be in an iteration in which event 1 does not occur, all queries in the intersection of both computations – $F^{\mathcal{T}}(x^*, y^*)$ and $F^{\widehat{\mathcal{T}}}(x, y)$ – were already in $Q(A)$ in the beginning of the iteration. Hence, $\widehat{\mathcal{G}}$ was chosen to be consistent with them, and $\widetilde{\mathcal{G}}$ is thus well-defined.

We can now define $\widetilde{\mathcal{P}}$ in a similar manner. For every query $u = (pk, x', r)$ such that $|x'| \leq n'$ (i.e., queries that were learned during the learning phase), $\widetilde{\mathcal{P}}(u)$ is defined consistently with $\mathcal{P}$. For every query $u = (x', r)$ such that $|x'| > n'$:

- If u is asked during the evaluation of $F^{\mathcal{T}}(x^*, y^*)$, we let $\widetilde{\mathcal{P}}(pk, x', r) = \mathcal{P}(pk, x', r)$.
- If during the evaluation of $F^{\mathcal{T}}(x^*, y^*)$, there exists some query $(td, \{(r_i, b_i)\}_i)$ to $\mathcal{P}^{-1}$ whose answer is x' and such that $pk = \mathcal{G}(td)$, then we set $\widetilde{\mathcal{P}}(pk, x', r_i) = \mathcal{P}(pk, x', r_i)$ for every i.
- If u is asked during the evaluation of $F^{\widehat{\mathcal{T}}}(x, y)$, then we let $\widetilde{\mathcal{P}}(pk, x', r) = \widehat{\mathcal{P}}(pk, x', r)$.
- If during the evaluation of $F^{\widehat{\mathcal{T}}}(x^*, y^*)$, there exists some query $(td, \{(r_i, b_i)\}_i)$ to $\widehat{\mathcal{P}}^{-1}$ whose answer is x' and such that $pk = \widehat{\mathcal{G}}(td)$, then we set $\widetilde{\mathcal{P}}(pk, x', r_i) = \widehat{\mathcal{P}}(pk, x', r_i)$ for every i.
- If none of the above cases holds, we define $\widetilde{\mathcal{P}}(pk, x', r) = \langle x', r \rangle$.

Similarly to the case of $\widetilde{\mathcal{G}}$, all queries that appear in both the aforesaid computations were learned by the adversary in previous iterations, and are defined consistently in both of them. What is still left to be shown in order to show that $\widetilde{\mathcal{P}}$ is well defined and is a valid oracle in $\mathfrak{S}$ is that the following two conditions are met:

1. There is no conflict (with respect to $\widetilde{\mathcal{G}}$) between the definition of $\widetilde{\mathcal{P}}$ via a query to $\mathcal{P}$ and a query to $\mathcal{P}^{-1}$ made in one or more of the computations $F^{\mathcal{T}}(x^*, y^*)$ and $F^{\widehat{\mathcal{T}}}(x, y)$.
2. For every $n \in \mathbb{N}$ and for every $pk \in \{0,1\}^n$, the predicate $\mathcal{P}_n(pk, \cdot, \cdot)$ is $(n, 2^{-n/3})$-balanced.

In order to see that condition (1) holds, suppose toward contradiction that such a conflict exists with respect to some pair of keys $pk = \widetilde{\mathcal{G}}(td)$. Note that this in particular means that td is an oracle query made by at least one of the computations (as we assumed that other trapdoors are mapped by $\widetilde{\mathcal{G}}$ to a public-key that does not appear in either computation), and consider the following cases:

- $\mathcal{P}(pk,x,r)=b$ and $\mathcal{P}^{-1}(td,\{\ldots,(r,1-b),\ldots\})=x$ both appear in the evaluation of $F^{\mathcal{T}}(x^*,y^*)$. Since the true oracle is of course consistent, this means that $\mathcal{G}(td)\neq pk$ and that $\widehat{\mathcal{G}}(td)=pk$ appears in the computation $F^{\widehat{\mathcal{T}}}(x,y)$. But since we assumed that when F queries $\mathcal{P}^{-1}(td,\{\ldots,(r,1-b),\ldots\})=x$ it also queries $\mathcal{G}(td)$, which means that this is a new query made by $F^{\mathcal{T}}(x^*,y^*)$ that A learns during the update step, in contradiction to our assumption that event 1 of Claim 4.4 does not occur.

- $\widehat{\mathcal{P}}(pk,x,r)=b$ and $\widehat{\mathcal{P}}^{-1}(td,\{\ldots,(r,1-b),\ldots\})=x$ both appear in the evaluation of $F^{\widehat{\mathcal{T}}}(x,y)$. This yields a contradiction similar to the last case, noting that $\widehat{\mathcal{T}}$ is chosen to be consistent and that the adversary also queries $\mathcal{G}$ with td.

- $\mathcal{P}(pk,x,r)=b$ appears in the evaluation of $F^{\mathcal{T}}(x^*,y^*)$ but $\widehat{\mathcal{P}}^{-1}(td,\{\ldots,(r,1-b),\ldots\})=x$ appears in $F^{\widehat{\mathcal{T}}}(x,y)$. But this means that in the update phase, the adversary also queries $\mathcal{G}$ with td. If $\widehat{\mathcal{G}}(td)\neq pk$, this means that $F^{\mathcal{T}}(x^*,y^*)$ also queries $\mathcal{G}(td)$ and that $\mathcal{G}(td)=pk$, which means that the adversary learns a new query $F^{\mathcal{T}}(x^*,y^*)$, yielding a contradiction. Otherwise, $\widehat{\mathcal{G}}(td)=\widetilde{\mathcal{G}}(td)=pk$ and the adversary queries $\mathcal{P}$ with (pk,x,r) in the update phase of the iteration. Again, since $\widehat{\mathcal{T}}$ was chosen to be a valid oracle, this means that $\widehat{\mathcal{P}}(pk,x,r)=1-b$, but since $\widehat{\mathcal{T}}$ is consistent with what A have learned about $\mathcal{T}$ so far, this means that this is the first time it queries $\mathcal{P}(pk,x,r)$, thus learning a new query made by $F^{\mathcal{T}}(x^*,y^*)$.

- $\widehat{\mathcal{P}}(pk,x,r)=b$ appears in the evaluation of $F^{\widehat{\mathcal{T}}}(x,y)$ but $\mathcal{P}^{-1}(td,\{\ldots,(r,1-b),\ldots\})=x$ appears in $F^{\mathcal{T}}(x^*,y^*)$. This again derives a contradiction similarly to the last case.

As for condition (2), we consider two cases of input lengths to $\widetilde{\mathcal{P}}$. Since for every $n''\leq n'$, we defined $\widetilde{\mathcal{P}}_{n''}=\mathcal{P}_{n''}$, and $\mathcal{P}_{n''}$ is $(n'',2^{-n''/3})$-balanced for every n'', $\widetilde{\mathcal{P}}_{n''}$ is also $(n'',2^{-n''/3})$-balanced for every $n''\leq n'$. For $n''>n'$, it is easy to see that had $\widetilde{\mathcal{P}}_{n''}$ been defined solely by the inner product $\widetilde{\mathcal{P}}_{n''}(pk,x',r)=\langle x',r\rangle$, it would have been $(n'',0)$-balanced for every pk. Since for every $pk\in\{0,1\}^n$, $x'\in\{0,1\}^{n''}$ it holds that $\widetilde{\mathcal{P}}_{n''}(pk,x',r)\neq\langle x',r\rangle$ for at most $2q(n)\cdot s(n)$ values of r, it follows that for every $pk\in\{0,1\}^n$ and $x',x''\in\{0,1\}^{n''}$:

$$\left|\Pr_{r\leftarrow\{0,1\}^{n''}}\left[\widetilde{\mathcal{P}}_{n''}(pk,x',r)=\widetilde{\mathcal{P}}_{n''}(pk,x'',r)\right]-\frac{1}{2}\right|\leq\frac{2q(n)\cdot s(n)}{2^{n''}}$$
$$\leq\frac{2^{2n''/3}}{2^{n''}}=2^{-n''/3}.$$

In other words, $\widetilde{\mathcal{P}}''_n$ is $(n'',2^{-n''/3})$-balanced for every n''.

We conclude that $\widetilde{\mathcal{T}}$ is a valid oracle for which $F^{\widetilde{\mathcal{T}}}(x^*,y^*)=F^{\widetilde{\mathcal{T}}}(x,y)=v^*$. If in addition, event 2 of Claim 4.4 does not hold, then $x\neq x^*$. Putting things together, we get that $F^{\widetilde{\mathcal{P}}}$ is not partially injective, deriving a contradiction. ■

4.2 $\mathcal{T}$ is One Way for Correlated Inputs

The proof that the oracle $\mathcal{T}$ is one way for correlated inputs (according to Definition 3.4) consists of the following two steps. First, we show that a uniformly-chosen predicate (not necessarily balanced) is one way with an extremely high probability. Then, we show that the uniform distribution over predicates is statistically close to the uniform distribution over balanced predicates (for our choice of $\ell(n)=n$ and $\delta(n)=2^{-n/3}$).

In more detail, recall that the trapdoor and public key in the experiment $\mathsf{Invert}_{\mathcal{T},A^{\mathcal{T}}}(n)$ are chosen as follows: First, the trapdoor td is chosen uniformly at random from the set $\{0,1\}^n$ and then the public key is set to be $pk = \mathcal{G}(td)$. Now, let R_n denote the uniform distribution over predicates mapping a triplet of strings of length n each, to an output bit (i.e., if $\mathcal{P}_n$ is a predicate drawn from R_n, then for every $pk \in \{0,1\}^n$ and $x, r \in \{0,1\}^n$ it holds that $\mathcal{P}(pk, x, r)$ is a uniformly-chosen bit which is independent of the value of $\mathcal{P}_n$ on all other inputs). The following lemma shows that when $\mathcal{P}_n$ is sampled from R_n, then any poly-query adversary inverts $\mathcal{P} = \{\mathcal{P}_{n'}\}_{n' \in \mathbb{N}}$ on inputs of length n (vis-à-vis Definition 3.4) with probability that is negligible in n, regardless of how $\mathcal{P}_{-n}$ is chosen (where we use $\mathcal{P}_{-n}$ to denote $\mathcal{P} \setminus \{\mathcal{P}_n\}$).

Lemma 4.5. *Let $q = q(n)$ be a function of the security parameter $n \in \mathbb{N}$. For any q-query algorithm A, any $n \in \mathbb{N}$ and any fixing of $\mathcal{P}_{-n}$, it holds that*

$$\Pr\left[\mathsf{Invert}_{\mathcal{T},A^{\mathcal{T}}}(n) = 1\right] \leq \frac{2q(n)}{2^n - q(n)}$$

where $\mathcal{P}_n \leftarrow R_n$.

Proof. Let (td^*, pk^*) be the trapdoor and public key chosen at the beginning of the experiment $\mathsf{Invert}_{\mathcal{T},A^{\mathcal{T}}}(n)$, let x^* be the uniform challenge value drawn in the experiment, and let x be A's output at the end of the experiment.

We first show that queries to $\mathcal{G}$ and to $\mathcal{P}^{-1}$ are of little use, since the probability of A finding a trapdoor td such that $\mathcal{G}(td) = pk^*$ is exponentially small. In more detail, assume without loss of generality that A always queries $\mathcal{G}$ with a trapdoor td before querying $\mathcal{P}^{-1}$ with a query that contains td. Denote by $Q_{\mathcal{G}}$ the set of queries made by A to $\mathcal{G}$, and denote by HIT the event in which there exists some $td \in Q_{\mathcal{G}}$ such that $\mathcal{G}(td) = pk^*$. Then, the following claim captures the above intuition.

Claim 4.6. $\Pr[\mathsf{HIT}] \leq q(n)/(2^n - q(n))$.

Proof of Claim 4.6. Denote by HIT_i the event that one of the first i queries of A was a query to $\mathcal{G}$ and was answered by pk^*. Then, before the ith query of A, and conditioned on $\overline{\mathsf{HIT}_{i-1}}$, it holds that $\Pr\left[td^* = td | \overline{\mathsf{HIT}_{i-1}}\right] = \Pr\left[td^* = td' | \overline{\mathsf{HIT}_{i-1}}\right]$ for any $td, td' \in \{0,1\}^n$ that were not previously queried by A (since td^* was uniformly sampled). Moreover, for every trapdoor $td \in \{0,1\}^n$ that was not already queried, it holds that $\Pr[\mathcal{G}(td) = pk^* | td \neq td^*] = 2^{-n}$. Hence, for all $i \in [q(n)]$ it holds that $\Pr\left[\mathsf{HIT}_i | \overline{\mathsf{HIT}_{i-1}}\right] \leq 1/(2^n - i + 1)$. Taking a union bound over all queries, we conclude the proof of the claim. ■

We now turn to prove that conditioned on $\overline{\mathsf{HIT}}$ (i.e., conditioned on A not finding a trapdoor corresponding to pk^*), it has very little chance of finding x^*. Denote by $Q_1, Q_2 \subseteq \{0,1\}^n$ the sets of arguments in the second coordinate of the queries made by A to $\mathcal{P}$ (i.e., the x's in A's queries), before and after x^* is drawn, respectively, and let $Q = Q_1 \cup Q_2$. Assume without loss of generality that A queries the oracle for $\mathcal{P}(pk^*, x, r)$ for some $r \in \{0,1\}^n$ (recall that x is A's output at the end of the experiment). Then, it holds that $\Pr\left[\mathsf{Invert}_{\mathcal{T},A^{\mathcal{T}}}(n) = 1\right] \leq \Pr[x^* \in Q]$. We first prove the following two claims.

Claim 4.7. $\Pr[x^* \in Q_1] \leq |Q_1|/2^n$.

Proof of Claim 4.7. x^* is chosen uniformly at random from $\{0,1\}^n$ while Q_1 is already fixed. Thus, x^* is drawn to be in Q_1 with probability $|Q_1|/2^n$. ■

Claim 4.8. $\Pr\left[x^* \in Q_2 | x^* \notin Q_1 \wedge \overline{\mathsf{HIT}}\right] \leq |Q_2| / \left(2^n - q(n)\right)$.

Proof of Claim 4.8. For each $x_i \in Q_2$ we let $Q(i)$ be the set of (second arguments in) queries issued by A before x_i was queried. The key observation is that once x^* is chosen and fixed, before asking a query containing x_i it holds that $\Pr\left[x' = x^* | x^* \notin Q(i) \wedge \overline{\mathsf{HIT}}\right] = \Pr\left[x'' = x^* | x^* \notin Q(i) \wedge \overline{\mathsf{HIT}}\right]$ for every $x', x'' \in \{0,1\}^n \setminus Q(i)$, where the probability is taken over the choice of $\mathcal{P}_n$ and x^*. Hence, for every $x_i \in Q_2$ it holds that $\Pr\left[x_i = x^* | x^* \notin Q(i) \wedge \overline{\mathsf{HIT}}\right] \leq 1/\left(2^n - |Q(i)|\right)$. Summing over all queries in Q_2 implies that

$$\Pr\left[x^* \in Q_2 | x^* \notin Q_1 \wedge \overline{\mathsf{HIT}}\right] \leq \sum_{i=1}^{|Q_2|} \frac{1}{2^n - |Q(i)|} \leq \frac{|Q_2|}{2^n - q(n)}.$$

■

Combining Claims 4.7 and 4.8 we obtain:

$$\begin{aligned}
\Pr\left[\mathsf{Invert}_{\mathcal{P},A^{\mathcal{P}}}(n) = 1\right] &\leq \Pr\left[x^* \in Q\right] \\
&\leq \Pr\left[x^* \in Q_1\right] + \Pr\left[x^* \in Q_2 |\, x^* \notin Q_1\right] \\
&\leq \Pr\left[x^* \in Q_1\right] + \Pr\left[x^* \in Q_2 |\, x^* \notin Q_1 \wedge \overline{\mathsf{HIT}}\right] + \Pr\left[\mathsf{HIT}\right] \\
&\leq \frac{|Q_1|}{2^n} + \frac{|Q_2|}{2^n - q(n)} + \frac{q(n)}{2^n - q(n)} \\
&\leq \frac{2q(n)}{2^n - q(n)}.
\end{aligned}$$

This concludes the proof of Lemma 4.5. ■

Next, we let B_n be the uniform distribution over predicates taking inputs in $\{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n$ that are $2^{-n/3}$-balanced for every fixing of pk in the first coordinate, and argue that it is statistically close to R_n.

Lemma 4.9. $\mathsf{SD}\left(R_n, B_n\right) \leq 2^{3n+1} \cdot \exp\left(-2^{n/3+1}/3\right)$.

Proof. Denote $\mathcal{P}_{pk} = \mathcal{P}_n(pk, \cdot, \cdot)$ for every $pk \in \{0,1\}^n$. We first prove that

$$\Pr_{\mathcal{P}_n \leftarrow R_n}\left[\exists pk \in \{0,1\}^n : \mathcal{P}_{pk} \text{ is not } 2^{-n/3}\text{-balanced}\right] \leq 2^{3n+1} \cdot \exp\left(-2^{n/3+1}/3\right).$$

Then, the proof is concluded by bounding the statistical distance by the aforementioned probability. Fix any $pk \in \{0,1\}^n$ and distinct $x, x' \in \{0,1\}^n$. Then, the probability over the choice of $\mathcal{P}_n \leftarrow R_n$ that $\mathcal{P}_{pk}$ is not $2^{-n/3}$-balanced with respect to x and x' can be bounded as follows:

$$\begin{aligned}
&\Pr_{\mathcal{P}_n \leftarrow R_n}\left[\left|\Pr_{r \leftarrow \{0,1\}^n}\left[\mathcal{P}_{pk}(x,r) = \mathcal{P}_{pk}(x',r)\right] - \frac{1}{2}\right| \geq 2^{-n/3}\right] \\
&\quad = \Pr_{\mathcal{P}_n \leftarrow R_n}\left[\left|\left|\left\{r : \mathcal{P}_{pk}(x,r) = \mathcal{P}_{pk}(x',r)\right\}\right| - 2^{n-1}\right| \geq 2^{2n/3}\right] \\
&\quad \leq 2 \cdot \Pr_{\mathcal{P}_n \leftarrow R_n}\left[\left|\left\{r : \mathcal{P}_{pk}(x,r) = \mathcal{P}_{pk}(x',r)\right\}\right| - 2^{n-1} \geq 2^{2n/3}\right] \\
&\quad \leq 2 \cdot \Pr_{\mathcal{P}_n \leftarrow R_n}\left[\left|\left\{r : \mathcal{P}_{pk}(x,r) = \mathcal{P}_{pk}(x',r)\right\}\right| \geq 2^{n-1}\left(1 + 2 \cdot 2^{-n/3}\right)\right] \\
&\quad \leq 2 \cdot \exp\left(-2^{n/3+1}/3\right).
\end{aligned}$$

where the last inequality follows by Chernoff bound. By taking a union bound over all pairs $x, x' \in \{0,1\}^n$ we obtain

$$\Pr_{\mathcal{P}_n \leftarrow R_n}\left[\mathcal{P}_{pk} \text{ is not } 2^{-n/3}\text{-balanced}\right] \leq 2^{2n+1} \cdot \exp\left(-2^{n/3+1}/3\right).$$

By another union bound, over all public keys $pk \in \{0,1\}^n$, we get

$$\Pr_{\mathcal{P}_n \leftarrow R_n}\left[\exists pk \in \{0,1\}^n : \mathcal{P}_{pk} \text{ is not } 2^{-n/3}\text{-balanced}\right] \leq 2^{3n+1} \cdot \exp\left(-2^{n/3+1}/3\right).$$

Finally, let $\mathcal{S}$ denote the set of all predicates taking three inputs of lengths n each (i.e., $\mathcal{S} = \{0,1\}^{\{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n}$), and let $\mathcal{B}$ be the set of all such predicates that are $2^{-n/3}$-balanced for every public key (i.e., $\mathcal{B} = \left\{\mathcal{P}_n \in \mathcal{S} : \mathcal{P}_{pk} \text{ is } 2^{-n/3}\text{-balanced for every } pk \in \{0,1\}^n\right\}$). Then, the statistical distance between B_n and R_n can be bounded as follows:

$$\begin{aligned}
\mathsf{SD}(R_n, B_n) &= \frac{1}{2} \cdot \left(\sum_{\mathcal{P}_n \in \mathcal{B}} |R_n(\mathcal{P}_n) - B_n(\mathcal{P}_n)| + \sum_{\mathcal{P}_n \in \mathcal{S} \setminus \mathcal{B}} |R_n(\mathcal{P}_n) - B_n(\mathcal{P}_n)|\right) \\
&= \frac{1}{2} \cdot \left(|\mathcal{B}| \cdot \left(\frac{1}{|\mathcal{B}|} - \frac{1}{|\mathcal{S}|}\right) + (|\mathcal{S}| - |\mathcal{B}|) \cdot \frac{1}{|\mathcal{S}|}\right) \\
&= 1 - \frac{|\mathcal{B}|}{|\mathcal{S}|} \\
&= \Pr_{\mathcal{P}_n \leftarrow R_n}\left[\exists pk \in \{0,1\}^n : \mathcal{P}_{pk} \text{ is not } 2^{-n/3}\text{-balanced}\right] \\
&\leq 2^{3n+1} \cdot \exp\left(-2^{n/3+1}/3\right).
\end{aligned}$$

■

4.3 Extension to Perfectly-Complete Key-Agreement Protocols

The attack presented in Section 4.1 can be modified to show that there is no black-box construction of a perfectly complete key-agreement protocol (in which the parties end up with the same key with probability 1) from any correlated-input balanced trapdoor predicate. This is again in contrast to the fact that there exists a non-black-box construction of a perfectly complete key-agreement protocol from a such a predicate (since such protocols can be constructed from any injective trapdoor function). The modified attacker gets as input the transcript T of such a protocol, and retrieves the key computed by the parties with probability 1. Note that even though secure key-agreement protocols imply one-way functions [IL89], it is not clear that they yield an injective (or a partially-injective) one-way function, even if they are perfectly complete. Thus, the separation does not immediately follow by that of Section 4.1.

Roughly speaking, following the outline laid out in [BKS+11, AS16a], the extension of the attack would go about as follows. Let (A, B) an black-box construction of a perfectly complete key-agreement protocol from the oracle $\mathcal{T}$, and let $q = q(n)$ denote a bound on the number of B's oracle queries. Then, the adversary will run $2q(n) + 1$ iterations, in each of which it will: (1) Choose a possible execution of A that is consistent with the transcript T and with what it has learned so far on the true oracle; (2) Compute A's resulted key k_i; (3) Update its knowledge of the true oracle similarly to the adversary of Section 4.1. Finally, it will output $k = \mathsf{majority}\{k_i\}_{i \in [2q(n)+1]}$. A similar argument to that of Claim 4.4 shows that by the perfect completeness of the protocol, in each iteration either the adversary learns a new oracle query made by B in the true execution, or k_i is the correct key computed by the parties. It follows that in the majority of the iterations (at least $q(n) + 1$), the adversary indeed computes the correct key.

Acknowledgements

We thank Iftach Haitner and Moni Naor for useful discussions in various stages of this work.

References

[ACR$^+$97] A. Andreev, A. Clementi, J. Rolim, and L. Trevisan. Weak random sources, hitting sets, and BPP simulations. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–272, 1997.

[ACR98] A. Andreev, A. Clementi, and J. Rolim. A new general derandomization method. *Journal of the ACM*, 45(1):179–213, 1998.

[AIK06] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36(4):845–888, 2006.

[And94] A. Andreev. Complexity of nondeterministic functions. *BRICS Report Series*, 1(2), 1994.

[AS15] G. Asharov and G. Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 191–209, 2015.

[AS16a] G. Asharov and G. Segev. Limits on the power of indistinguishability obfuscation and functional encryption. *SIAM Journal on Computing*, 45(6):2117–2176, 2016.

[AS16b] G. Asharov and G. Segev. On constructing one-way permutations from indistinguishability obfuscation. In *Proceedings of the 13th Theory of Cryptography Conference*, pages 512–541, 2016.

[AW14] B. Applebaum and E. Widder. Related-key secure pseudorandom functions: The case of additive attacks. Cryptology ePrint Archive, Report 2014/478, 2014.

[Bar01] B. Barak. How to go beyond the black-box simulation barrier. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.

[BBF13] P. Baecher, C. Brzuska, and M. Fischlin. Notions of black-box reductions, revisited. In *Advances in Cryptology – ASIACRYPT '13*, pages 296–315, 2013.

[BC10] M. Bellare and D. Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *Advances in Cryptology – CRYPTO '10*, pages 666–684, 2010.

[BDV17] N. Bitansky, A. Degwekar, and V. Vaikuntanathan. Structure vs. hardness through the obfuscation lens. In *Advances in Cryptology - CRYPTO '17*, pages 696–723, 2017.

[BGI$^+$12] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6, 2012.

[BHS$^+$98] M. Bellare, S. Halevi, A. Sahai, and S. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In *Advances in Cryptology – CRYPTO '98*, pages 283–298, 1998.

[BK03] M. Bellare and T. Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In *Advances in Cryptology – EUROCRYPT '03*, pages 491–506, 2003.

[BKS$^+$11] Z. Brakerski, J. Katz, G. Segev, and A. Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In *Proceedings of the 8th Theory of Cryptography Conference*, pages 559–578, 2011.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[BOV07] B. Barak, S. J. Ong, and S. P. Vadhan. Derandomization in cryptography. *SIAM Journal on Computing*, 37(2):380–400, 2007.

[BP12] N. Bitansky and O. Paneth. From the impossibility of obfuscation to a new non-black-box simulation technique. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 223–232, 2012.

[BPW16] N. Bitansky, O. Paneth, and D. Wichs. Perfect structure on the edge of chaos – trapdoor permutations from indistinguishability obfuscation. In *Proceedings of the 13th Theory of Cryptography Conference*, pages 474–502, 2016.

[BV17] N. Bitansky and V. Vaikuntanathan. A note on perfect correctness by derandomization. In *Advances in Cryptology – EUROCRYPT '17*, pages 592–606, 2017.

[CG89] B. Chor and O. Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5(1):96–106, 1989.

[CPS16] K. Chung, R. Pass, and K. Seth. Non-black-box simulation from one-way functions and applications to resettable security. *SIAM Journal on Computing*, 45(2):415–458, 2016.

[Dac16] D. Dachman-Soled. Towards non-black-box separations of public-key encryption and one-way function. In *Proceedings of the 14th Theory of Cryptography Conference*, pages 169–191, 2016.

[DG17] N. Döttling and S. Garg. Identity-based encryption from the Diffie-Hellman assumption. In *Advances in Cryptology – CRYPTO '17*, pages 537–569, 2017.

[DN07] C. Dwork and M. Naor. Zaps and their applications. *SIAM Journal on Computing*, 36(6):1513–1543, 2007.

[DNR04] C. Dwork, M. Naor, and O. Reingold. Immunizing encryption schemes from decryption errors. In *Advances in Cryptology – EUROCRYPT '04*, pages 342–360, 2004.

[GGH$^+$13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, pages 40–49, 2013.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In *Advances in Cryptology – CRYPTO '84*, pages 276–288, 1984.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GK96] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GL10] D. Goldenberg and M. Liskov. On related-secret pseudorandomness. In *Proceedings of the 7th Theory of Cryptography Conference*, pages 255–272, 2010.

[GLN11] O. Goldreich, L. A. Levin, and N. Nisan. On constructing 1-1 one-way functions. In *Studies in Complexity and Cryptography*, pages 13–25. Springer, 2011.

[GMM17] S. Garg, M. Mahmoody, and A. Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In *Advances in Cryptology – CRYPTO '17*, pages 661–695, 2017.

[GMW86] O. Goldreich, S. Micali, and A. Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *Advances in Cryptology – CRYPTO '86*, pages 171–185, 1986.

[Gol01] O. Goldreich. Foundations of Cryptography – Volume 1: Basic Techniques. Cambridge University Press, 2001.

[Gol11] O. Goldreich. In a world of P= BPP. In *Studies in Complexity and Cryptography*, pages 191–232. Springer, 2011.

[GSTS03] D. Gutfreund, R. Shaltiel, and A. Ta-Shma. Uniform hardness versus randomness tradeoffs for Arthur-Merlin games. *Computational Complexity*, 12(3):85–130, 2003.

[GVW11] O. Goldreich, S. P. Vadhan, and A. Wigderson. Simplified derandomization of BPP using a hitting set generator. In *Studies in Complexity and Cryptography*, pages 59–67. Springer, 2011.

[HIL$^+$99] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HNO$^+$09] I. Haitner, M. Nguyen, S. J. Ong, O. Reingold, and S. P. Vadhan. Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. *SIAM Journal on Computing*, 39(3):1153–1218, 2009.

[IL89] R. Impagliazzo and M. Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 230–235, 1989.

[Imp95] R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Structure in Complexity Theory Conference*, pages 134–147, 1995.

[IR89] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61, 1989.

[IW97] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215–217, 1983.

[LLS$^+$97] N. Linial, M. Luby, M. Saks, and D. Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Combinatorica*, 17(2):215–234, 1997.

[LMR14] K. Lewi, H. Montgomery, and A. Raghunathan. Improved constructions of PRFs secure against related-key attacks. In *International Conference on Applied Cryptography and Network Security*, pages 44–61, 2014.

[LR88] M. Luby and C. Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[Luc04] S. Lucks. Ciphers secure against related-key attacks. In *International Workshop on Fast Software Encryption*, pages 359–370, 2004.

[MM11] T. Matsuda and K. Matsuura. On black-box separations among injective one-way functions. In *Proceedings of the 8th Theory of Cryptography Conference*, pages 597–614, 2011.

[MP12] M. Mahmoody and R. Pass. The curious case of non-interactive commitments – On the power of black-box vs. non-black-box use of primitives. In *Advances in Cryptology – CRYPTO '12*, pages 701–718, 2012.

[MV99] P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 71–80, 1999.

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

[NR99] M. Naor and O. Reingold. On the construction of pseudorandom permutations: Luby-Rackoff revisited. *Journal of Cryptology*, 12(1):29–66, 1999.

[NW94] N. Nisan and A. Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 33–43, 1989.

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

[PRV12] P. A. Papakonstantinou, C. A. Rackoff, and Y. Vahlis. How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653, 2012.

[PTV11] R. Pass, W. D. Tseng, and M. Venkitasubramaniam. Towards non-black-box lower bounds in cryptography. In *Proceedings of the 8th Theory of Cryptography Conference*, pages 579–596, 2011.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, 1990.

[RSS17] A. Rosen, G. Segev, and I. Shahaf. Can PPAD hardness be based on standard cryptographic assumptions? In *Proceedings of the 15th Theory of Cryptography Conference*, pages 747–776, 2017.

[RTV04] O. Reingold, L. Trevisan, and S. P. Vadhan. Notions of reducibility between cryptographic primitives. In *Proceedings of the 1st Theory of Cryptography Conference*, pages 1–20, 2004.

[Rud88] S. Rudich. Limits on the Provable Consequences of One-way Functions. PhD thesis, EECS Department, University of California, Berkeley, 1988.

[Sim98] D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT '98*, pages 334–345, 1998.

[Sip88] M. Sipser. Expanders, randomness, or time versus space. *J. Comput. Syst. Sci.*, 36(3):379–383, 1988.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.

A Constructing Correlated-Input Predicates from Related-Secret Primitives

In section 1.2 we observed that a correlated-input balanced one-way predicate can be constructed directly from any injective one-way function. In this section we additionally show how such predicates can be constructed from various pseudorandom primitives that are secure against related secret attacks (and that do not seem to guarantee any form of injectivity to begin with). The most prominent example of such primitives are pseudorandom functions secure against related-key attacks (RKAs), formally defined by Bellare and Kohno [BK03]. Loosely speaking, in the setting of PRFs secure against RKAs with respect to some function family Δ, once the PRF key k is chosen, the adversary is allowed to ask for evaluations of the oracle to which it has access under keys related to k via any function $\delta \in \Delta$. Canonical examples are those of "linear nature", which include the families of key shifts (e.g. via an exclusive or operation or modular addition), and group induced-transformations (where the related keys are the result of applying the group operation on the key and an additional element). In fact, for our needs we only need to consider a week form of related-secret attacks, were the adversary must choose its queries *non-adaptively.*

PRFs secure against non-trivial *adaptive* key transformations were first shown to exist in idealized models. Namely, Bellare and Kohno showed in [BK03] that an ideal cipher is a PRF secure against related key attacks with respect to families of functions exhibiting some intuitive properties, including key shifts, and Lucks [Luc04] observed that if the key to a PRF is hashed via a random oracle before use then the PRF becomes secure against many families of key transformations. A subsequent line of works gave rise to constructions of RKA-secure PRFs from concrete assumptions. Most notably, Bellare and Cash [BC10] constructed such PRFs secure against group-induced key transformations based on the DDH assumption (and PRFs based on the DLIN assumption as well). Lewi et al. [LMR14] managed to construct PRFs secure against linear key transformations based on lattices, and namely the learning with errors (LWE) assumption. In the passive setting, where the key transformations are chosen randomly and not by the adversary, Applebaum and Widder [AW14] constructed an RKA-secure PRF from the learning with rounding (LWR) assumption. They then continued to show that in "MiniCrypt" — where one-way functions exist but public key encryption does not — PRFs secure against passive RKA-attacks can be transformed to PRFs secure against non-adaptive RKA-attacks (where the adversary may choose the key transformations, but must do so in advance). All of the above constructions can be used to instantiate a construction of correlated-input balanced one-way predicates in ways similar to what we show below.

Goldenberg and Liskov [GL10] relaxed the notion of RKA-secure PRFs and defined pseudorandom generators secure against related-secret attacks, which they showed to be equivalent to RKA-secure PRFs. Thus, for the sake of simplicity, we concentrate on showing how to construct correlated-input balanced one-way predicates from PRGs secure against related secret attacks, where similar constructions trivially exist from RKA-secure PRFs. Furthermore, though Goldenberg and Liskov defined such PRGs in the adaptive setting, we will restrict ourselves to the simpler, yet sufficient for our needs, non-adaptive case. Informally speaking, an efficiently computable function G is a PRG secure against non-adaptive related secret attacks with respect to a function family Δ, if an efficient adversary cannot distinguish between G and a truly random function, even when it may ask in advance for evaluations of $G(\delta(s))$ for any $\delta \in \Delta$ (where, as usual, s is chosen uniformly at random)[10].

Definition A.1. Let $G : \{0,1\}^* \to \{0,1\}^*$ be an efficiently computable function which maps n bits to $\ell(n)$ bits for every $n \in \mathbb{N}$. We say G is a *pseudorandom generator secure against non-adaptive related-secret attacks* with respect to a function family Δ, if for every probabilistic polynomial-time adversary A there exists a negligible function $\nu(\cdot)$ such that

$$\mathsf{Adv}_{G,A,\Delta}(n) \overset{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}_{G,A,\Delta}(n) = 1\right] - \Pr\left[\mathsf{Exp}_{\mathcal{O},A,\Delta}(n) = 1\right]\right| \leq \nu(n)$$

for all sufficiently large $n \in \mathbb{N}$, where $\mathcal{O}$ is a randomly drawn function mapping n bits to $\ell(n)$ bits and for $H \in \{G, \mathcal{O}\}$ the experiment $\mathsf{Exp}_{H,A,\Delta}(n)$ is defined as follows:

1. $(\mathsf{state}, \delta_1, \ldots, \delta_T) \leftarrow A(1^n)$ for $\delta_1, \ldots, \delta_T \in \Delta$, where $T = T(n)$ may be any polynomial determined by A.
2. $b \leftarrow A(\mathsf{state}, H(\delta_1(s)), \ldots, H(\delta_T(s)))$ where $s \leftarrow \{0,1\}^n$.
3. output b.

For concreteness, we consider PRGs secure against non-uniform adversaries making non-adaptive related-secret queries with respect to the set of key shifts, $\Delta_\oplus = \left\{\delta_r(x) = x \oplus r : r \in \{0,1\}^{|x|}\right\}$. We observe that such a PRG immediately yields a correlated-input (n, ν)-balanced one-way predicate for some negligible function $\nu(\cdot)$. Let G be such a PRG, and define $\mathcal{P}(x, r)$ to return the parity bit of $G(x \oplus r)$. The correlated-input one-wayness of $\mathcal{P}$ is guaranteed by the related-secret pseudo-randomness of G. Moreover, suppose that $\mathcal{P}$ is not balanced; meaning, there are x, x' and a polynomial $p(\cdot)$, such that

$$\left|\Pr_{r \leftarrow \{0,1\}^n}\left[\text{Parity}\left(G(x \oplus r)\right) = \text{Parity}\left(G(x' \oplus r)\right)\right] - \frac{1}{2}\right| > 1/p(n).$$

In that case, an adversary can simply query its oracle with x and x' (which it gets as non-uniform advice) and compare the parity bits of the results. In case $H = G$, then the two bits will be identical with probability that differs from $1/2$ by at least $1/p(n)$. On the other hand, if the $H = \mathcal{O}$, then the two answers will be the same with probability exactly $1/2$, thus allowing the adversary to distinguish between the cases with noticeable advantage.

[10] Unlike typical PRGs, a PRG secure against related secret attacks is not necessarily length increasing, and in fact, we do not need it to be for our needs.

Note that the construction based on related-secret PRGs yields a primitive with properties that are potentially stronger than those specified by Definition 1.1: The vector

$$(\mathcal{P}(x, r_1), \ldots, \mathcal{P}(x, r_T)) = (\text{Parity}\,(G(x \oplus r_1))\,, \ldots, \text{Parity}\,(G(x \oplus r_T)))$$

given to the adversary in step 2 of the security experiment of Definition 1.1 is not only one-way with respect to x, but is pseudorandom. Indeed, in our construction in Section 3, if we use $\mathcal{P}$ with this strengthen property, we get an injective PRG, and not just an injective one-way function.