On the Construction of One-Way Functions from Average Case Hardness

Noam Livne
Weizmann Institute of Science, Rehovot, Israel
noam.livne@weizmann.ac.il

Abstract: In this paper we study the possibility of proving the existence of one-way functions based on average case hardness. It is well-known that if there exists a polynomial-time sampler that outputs instance-solution pairs such that the distribution on the instances is hard on average, then one-way functions exist. We study the possibility of constructing such a sampler based on the assumption that there exists a sampler that outputs only instances, where the distribution on the instances is hard on the average. Namely, we study the possibility of "modifying" an ordinary sampler S that outputs (only) hard instances of some search problem R, to a sampler that outputs instance-solution pairs of the same search problem R. We show that under some restriction, not every sampler can be so modified. That is, we show that if some hard problem with certain properties exists (which, in particular is implied by the existence of one-way permutations), then for every polynomial λ, there exists a search problem R and a polynomial-time sampler S such that (1) R is hard under the distribution induced by S, and (2) there is no sampler S^* with randomness complexity bounded by λ, that outputs instance-solution pairs of R, where the distribution on the instances of S^* is closely related to that of S (i.e., dominates it). A possible interpretation of our result is that a generic approach for transforming samplers of instances to samplers of instance-solution pairs cannot succeed.

Keywords: One-Way Function; Average-Case Hardness

1 Introduction

The existence of one-way functions is a necessary condition for nearly all cryptographic applications, and is sufficient for a large portion of them. Since establishing the existence of one-way functions does not seem reachable these days, as it implies that $\mathcal{P} \neq \mathcal{NP}$, a line of papers studied the possibility of proving the existence of one-way functions based on the assumption that $\mathcal{P} \neq \mathcal{NP}$ ([3],[4],[2],[1]). These works presented limitations on possible reductions of the security of one-way functions to $\mathcal{NP}$-hardness. Specifically, these works show that, under certain assumptions, certain reductions of this type do not exist.

The starting point of this paper is the observation that the transformations studied by these papers are actually supposed to overcome two gaps at once: (1) the gap between average-case hardness and worst-case hardness; and (2) the gap between one-way functions and average-case hardness, where "average case hardness" refers to a search problem R and a sampler S such that the search problem R is hard on average under the distribution induced by S. Since the problem of basing average-case hardness on worst-case hardness is hard by itself, it is inviting to study the seemingly more modest task of basing one-way functions on average-case hardness.

The assumption that average case hardness exists (as defined above), implies the existence of objects (namely, R and S as above) one can hope to base a construction of a one-way function upon (while to the best of our understanding, the assumption that $\mathcal{P} \neq \mathcal{NP}$ does not seem to imply such objects). Indeed, this paper concentrates on the question of *constructing* a one-way function from average case hardness, rather than on *proving the security* of one-way functions based on average case hardness. Our result gives limitations on a certain approach to achieve this construction. Informally, our result implies that (under some restriction), this approach cannot work for any R and S as above.

1.1 Main Result

Before describing our result, let us describe the approach we refer to above. How can one base the construction of a one-way function on the existence of average case hardness? As aforesaid, the assumption implies a search problem R and a polynomial-time sampler S where the search problem R is hard under the distribution induced by S. A natural way to go is to use the following known fact, stated informally (for more details see [5], Section 7.1.1.):

Observation 1.1 *If there exists a search problem*

R and a polynomial-time sampler S^* that outputs instance-solution pairs of R, such that the distribution of S^* restricted to the instances is hard on average, then one-way functions exist.

Using this fact, one can try to construct a new sampler S^*, that "behaves" similarly to S, but instead of outputting only instances, S^* outputs instance-solution pairs (under R). That is, the distribution induced by S^*, when restricted to the instances only, is similar to the distribution of S. Thus, the sampler S^* "inherits" its hardness from S. In fact, in order for S^* to inherit the hardness of S, it is sufficient that the (restricted) distribution of S^* dominates that of S (see Section 2 for definitions). Upon constructing such S^*, one can use Observation 1.1 to construct the one-way function. Thus, this approach of basing one-way functions on average case hardness calls for the following challenge:

Desired Construction:
Given a search problem R and a polynomial-time sampler S, where R is hard under the distribution induced by S, construct a polynomial-time sampler S^* that outputs instance-solution pairs of R, such that the distribution of S^* restricted to the instances, dominates the distribution of S.

Informally, our main result is that for any fixed polynomial bounding the randomness complexity of S^*, the Desired Construction cannot always exist. That is, under plausible assumptions (which in particular are implied by the existence of one-way permutations), for any polynomial λ there exists a search problem R and a polynomial time sampler S such that R is hard under S, but where a sampler S^* as in the Desired Construction with randomness complexity bounded by λ, does not exist.

We note, that the search problem R we construct is polynomially bounded (i.e., the lengths of the solutions are polynomially bounded in the lengths of the instances), but not polynomial time verifiable. Rather, it can be verified in arbitrarily small super-polynomial time. This still leaves open the possibility that for any R and S, where R is polynomial time verifiable, the Desired Construction exists (even for some universal polynomial randomness complexity of S^*). However, we note that in order for the Desired Construction to yield a one-way function, there is no need that R be polynomial time verifiable. Thus, one might hope that the construction exists also when R is not polynomial time verifiable. While we cannot eliminate this possibility, we can give arbitrarily strong polynomial lower bounds on the randomness complexity of S^* that achieves this construction. (For a discussion on this issue see Section 4.2.)

We also note that placing no restrictions on R and S, the result is trivial (even without fixing the randomness complexity of S^*):

- If the lengths of the solutions in R is not polynomially bounded in the lengths of the instances, then trivially no such S^* exists simply because it cannot write the solutions in polynomial time.
- If the lengths of the solutions in R is polynomially bounded in the lengths of the instances, but R is super-exponentially (worst-case) hard, then again no such S^* exists: take S to be an "onto sampler" (i.e., a sampler that outputs any instance with positive probability). Then the assumption that a sampler S^* as above exists implies that R can be solved in the worst case in exponential time, in contradiction: Given some instance x, one can find an instance-solution pair where the instance is x, by performing an exhaustive search on the randomness for S^* until it outputs an appropriate pair (notice that S^* outputs for every x a pair where the instance is x, since the distribution on the instances in S^* should dominate S).

While for the first case above such R and S can be constructed (simply by padding) from the basic assumption that hardness on average exists (as defined above), the second case seems to require a stronger assumption regarding the hardness on average.

1.2 Technique

In the following we explain informally the idea of our proof. Suppose, as a mental experiment, that for some R and S there *does exist* S^* as in the Desired Construction. Then, although R is hard under S, using S^* one can sample *random* instance-solution pairs of R. Intuitively, given a specific instance one cannot find a solution, but one can always easily find a *random* instance together with a solution. Moreover, one can obtain instance-solution pairs that relate to arbitrary coin-vectors for S^* at his choice. In light of these observations, the basic idea in our proof is the following. Given some R', S', we construct R and S such that:

- If R' is hard under S' then R is hard under S (that is, the pair (R, S) "inherit" the hardness of the pair (R', S')). This is achieved by "embedding" R' in R and S' in S. That is, any instance of R' is embedded in some instance of R, and any solution for an instance x of R "embeds" a solution for the instance of R' that is embedded in x. Moreover, roughly speaking, S imposes the distribution of S' on the embedded instances of R'.

- If an S^* exists for (R, S) then, if $(x, y) \in R$, the solution y helps finding pairs $(\hat{x}, \hat{y})$ for $\hat{x}$'s that are "close" to x. That is, for any $\hat{x}$ that is a neighbour of x under some (Hamming-like) metric, the solution y contains a coin-vector for S^* that yield an instance-solution pair of the form $(\hat{x}, \hat{y})$. Thus, on input a specific instance x for which one wants to find a solution, through repeated invocations of S^*, one can start from an arbitrary instance-solution pair, and go through a sequence of pairs, where the instance in each pair is closer to x than the preceding one by one unit under the aforementioned metric, until reaching x. By choosing an appropriate coin-vector for S^* from the solution in each pair, one can indeed have the instance in the next pair closer to x.

Thus, the assumption that S^* exists for (R, S), yields that it is easy to find an instance-solution pair with respect to R, for any *specific* instance. It follows that R is easy in the worst case. Now, since R "embeds" R', it follows that R' is also easy in the worst case, in contradiction.

It should be noted, however, that in order for this idea to work, some technicalities need to be dealt with, and these yield some restrictions on R' and S' (but, as aforementioned, in particular, the assumption that one-way permutations exist yields R' and S' as required for our proof).

2 Preliminaries

Standard Notation and Conventions For $1 \le i \le \ell$ we define $e_i^\ell \triangleq 0^{i-1}10^{\ell-i}$. Given a TM M we denote by $\langle M \rangle$ the code of M. We say that a function f is *noticeable* if there exists a positive polynomial q such that for large enough n's, $f(n) \ge 1/q(n)$. Given a string x we denote by $|x|$ the length of x. We use interchangeably the terms *search problem* and *relation.* Given a search problem R we say that *x is a YES-instance of* R if x has a solution under R, that is, if there exists y such that $(x, y) \in R$. We say that a search problem is *total* if all strings are YES-instances of that problem. We say that a search problem is *polynomially bounded* if there exists some polynomial q such that for every y which is a solution for x, it holds that $|y| \le q(|x|)$. When defining strings in the form $(\cdot)$, $(\cdot, \cdot)$ etc., we implicitly assume some 1-1, onto, efficient, efficiently invertible encoding[1] from $\bigcup_{n \in \mathbb{N}} (\Sigma^*)^n$ to Σ^*. We will use the term *support* in relation to an ensemble of random variables to denote the sequence of supports of the variables in the ensemble.

[1]For simplicity we assume $|(x, y)| = |x| + |y|$. While this is not the case, it will make the presentation clearer. Clearly, this technicality can be handled, without any essential change in the statements and proofs.

In our proof we would like to enumerate all samplers. However, since any reasonable definition of a sampler assumes some structural properties (for example, to the least the requirement that it always halt), it is not clear how one can enumerate all samplers. Instead, we enumerate a superset of the set of samplers, which we call *potential samplers*, which are basically just probabilistic TM's.

Definition 2.1 (Potential Sampler) *A potential sampler is a TM with 2 input tapes. The input to the first tape is simply called "input", and the input to the second tape is called "randomness".*

We implicitly assume some enumeration on all such machines. We explain how we interpret a potential sampler. The run of a potential sampler on some input, when not stating explicitly the randomness, is defined as a random variable (ranging over all strings plus a special "not halting" symbol), which is the result of running the machine on a random infinite vector of coins (equivalently, one can think of the machine as flipping coins on the fly). We define the output of a potential sampler given some input and some *explicit* randomness as the output on that input and randomness, and for sake of completeness, we define the output in the case the sampler requires more coins than the explicit randomness, as the failure symbol $\perp$.

Definition 2.2 (Sampler) *A sampler is a potential sampler that on input 1^n, with probability 1 halts and outputs a string of length n.*

We explain how we interpret a sampler. Typically we will run a sampler on inputs of the form 1^n (which will be referred to as "input"). In such a case, for a sampler S, the term $S(1^n)$ will be a random variable with values from Σ^n. Thus, any sampler S defines an ensemble of random variables $\{S(1^n)\}_{n \in \mathbb{N}}$. When needed, we will relate explicitly to the randomness as (a second) input to the sampler.

We note, that our rigorous definition of a sampler is for sake of formality of the proof, and is arbitrary. To the best of our knowledge, if there exists a sampler under any "reasonable" definition that generates hard instances of R, then there exists a sampler under our definition that generates hard instances of R (see definition of hardness below).

Definition 2.3 (Onto Sampler) *A sampler is an onto sampler if it outputs every string with positive probability. Formally, a sampler is an onto sampler if its support is $\{\{0,1\}^n\}_{n \in \mathbb{N}}$.*

Definition 2.4 (Randomness Complexity of a Potential Sampler) *We say that the function f is the randomness complexity of some potential sampler, if on input 1^n, the maximal number of coins it uses is $f(n)$.*

We also define the following transformation, that transforms any machine M that output pairs of strings, to a machine $\tilde{M}$ such that on any input and randomness for which M outputs a pair, $\tilde{M}$ outputs only the first element in the pair:

Unpairing Transformation:

On input $\langle M \rangle$ (a code of a potential sampler), the transformation outputs the code of the following potential sampler $\tilde{M}$:

On input x and randomness r, the machine $\tilde{M}$ runs M on x with randomness r, if it halts it checks if the output is a pair, and if it is, it deletes the second element in the pair.

It is easy to see that the transformation is efficiently computable and that the running time of $\tilde{M}$ is linearly related to that of M. From now on, given a potential sampler M, we denote by $\tilde{M}$ the potential sampler who's code is the result of applying the transformation on $\langle M \rangle$.

Definition 2.5 (Pairs-Sampler) *A pairs-sampler is a potential sampler M that on every input and randomness outputs a pair, and where $\tilde{M}$ is a sampler according to Definition 2.2 (that is, on input 1^n, with probability 1 M halts and outputs a pair where the first element is of length n).*

Throughout the paper we consider search problems that are not total, along with samplers that may not output only YES-instances (for these search problems). The following definition relates to this setting. It states that in the aforementioned setting, a potential solver A fails on a search problem R under the sampler S if when running A on random instances output by S, the event that a YES-instance is output and A does not respond with a correct solution, is noticeable. (Trivially, it implies that S must output YES-instances with noticeable probability.) We then say that R is hard under S if every potential probabilistic polynomial time solver A fails on R under S.

Definition 2.6 (Failure of a Solver, Hardness of a Search Problem Under a Sampler) *For an algorithm A, a search problem R and a sampler S we say that A* fails *on R* under *S if:*

$$\Pr_{x \leftarrow S(1^n)}[(x, A(x)) \notin R \wedge \exists y (x, y) \in R]$$

is noticeable (where the probability is taken both over the randomness of S, and the randomness of A).

We say that R is hard under *S if every probabilistic polynomial time algorithm fails on R under S.*

We note that one can alternatively define the failure of a solver, instead of failing with noticeable probability (on the aforementioned distribution), as not failing with negligible probability. The difference is that now we only require that the solver fail infinitely often, and not on every n. Our main result is true also under that definition, and the changes in the proof are minor.

Definition 2.7 (Domination) *Given two ensembles of random variables $\{X_n\}_{n\in\mathbb{N}}$ and $\{X'_n\}_{n\in\mathbb{N}}$, we say that $\{X_n\}$* dominates *$\{X'_n\}$ if there exists a positive polynomial q such that for any n and any $x \in X'_n$, $\Pr[X_n = x] \geq (1/q(n)) \Pr[X'_n = x]$.*

As aforementioned, every sampler induces an ensemble of random variables. For brevity, sometimes we will say that a sampler S dominates a sampler S' to mean that the ensemble induced by S dominates that of S'.

The following facts, which will be used through the paper, are straightforward:

Fact 2.8 *If a sampler S dominates a sampler S' and S' is an onto sampler then so is S.*

Fact 2.9 *Let $\{X_n\}_{n\in\mathbb{N}}$ be an ensemble that dominates the ensemble $\{X'_n\}_{n\in\mathbb{N}}$ and let $\{S_n\}_{n\in\mathbb{N}}$ be a sequence of sets. Suppose $\Pr[X'_n \in S_n]$ is noticeable. Then $\Pr[X_n \in S_n]$ is noticeable.*

3 Main Result

Theorem 3.1 *Suppose that there exists a polynomially bounded total search problem R' that is hard under some polynomial time onto sampler. Then, for any super-polynomial, time-constructible function τ, and every polynomial $\lambda(n) \geq n$, there exists a search problem R and a polynomial time sampler S such that:*

- *R is hard under S.*
- *There is no polynomial time pairs-sampler S^* with randomness complexity bounded by λ, such that the first elements in the pairs output by S^* are distributed in a distribution that dominates $\{S(1^n)\}_{n\in\mathbb{N}}$, and the second element in every pair is a solution for the first element with respect to R if such a solution exists, and any string otherwise.*
- *The relation R is polynomially bounded, and can be verified in time $\tau(n)$ using one oracle call to a verifier for R'.*

We note that the resulted search problem R is not total, and that the sampler S does not output only YES-instances. Nevertheless, the existence of a pairs-sampler S^* as (asserted not to exist) in the theorem for such R and S implies the existence of one-way functions.

Proof: Let R' be a search problem that is hard under the polynomial time sampler S', from the hypothesis. Let τ be some super polynomial time-constructible function (for sake of concreteness one can think of $\tau(n) = n^{log^*(n)}$). Let τ' be a (smaller) super polynomial time-constructible function to be defined later. Let λ be a polynomial. We define S and R as follows:

Definition of R:
$((\langle M\rangle, x), (w, r_1, \ldots, r_{|x|})) \in R$ if and only if the following conditions hold:

- $(x, w) \in R'$
- For all $1 \leq i \leq |x|$ it holds that $|r_i| \leq \lambda(|\langle M\rangle| + |x|)$.
- For all $1 \leq i \leq |x|$, on input $1^{|\langle M\rangle|+|x|}$ and randomness r_i, the potential sampler $\tilde{M}$ outputs $(\langle M\rangle, x \oplus e_i^{|x|})$ in at most $\tau'(|\langle M\rangle| + |x|)$ steps.

In the following we describe the main ideas behind the definitions (disregarding issues of running time and randomness complexity, which will be dealt later). Suppose that M is some potential sampler (we will eventually consider $M = S^*$). Then, if $(w, r_1, \ldots, r_{|x|})$ is a solution for $(\langle M\rangle, x)$, then the r_i's, when used as randomness for M (with the appropriate input length), output an instance-solution pair where in the instance, the first element (the machine code) is again the code of the same machine M, and the second element in the instance is x with the i-th bit flipped. Thus, using the appropriate r_i from the solution as randomness for M, one can basically "flip any bit of x at his choice" without changing the (code of) the machine in the instance. Now, suppose the machine M is guaranteed to always output legal pairs with respect to R, when the instance in the pair is a YES-instance. Then, if the new instance (with the "flipped" bit) is a YES-instance, the new pair output by M is again a legal pair. It follows, that if all involved instances are YES-instances, then by repeated invocations of M (each time using the appropriate r_i from the last solution), one can sequentially flip bit after bit, and arrive at any desired x'. We then note that if $(w, r_1, \ldots, r_{|x|})$ is a solution for $(\langle M\rangle, x)$ under R, then w is a solution for x under R'. Thus, once arriving at such x' (i.e., once having M output a pair $((\langle M\rangle, x'), (w, r_1, \ldots, r_{|x|}))$, indeed w is a solution for x' under R'. It follows that R' can be efficiently solved (in the worst case), in contradiction to the hardness of R' under S'.

The following definition of S, together with the hypotheses of the theorem, guarantee that if a sampler S^* exists, the idea outlined above works. It also guarantees that R is hard under S, as required. We elaborate. Since we would like to diagonalize against all possible S^*'s, roughly speaking, we let S output (the code of) any machine (in the left part of the instance) with noticeable probability (see Claim 3.2). This implies (by the hypothesis of the theorem) that any potential S^* must also output (the code of) any machine with noticeable probability (as it is required to dominate S). If follows that one can efficiently (i.e., in expected polynomial time), find randomness for S^* that yield any desired machine. In particular, one can efficiently find randomness for S^* that will make it output *its own code* (and again, this is true for any possible S^*). Since S^* is assumed to output only legal pairs with respect to R when these exist, the process described above can then be performed, provided that legal solutions always exist. We will show that Since $\tilde{S^*}$ must be an onto-sampler (since S is an onto sampler and $\tilde{S^*}$ dominates it), and since R' is total, throughout the process described above solutions indeed will always exist. Moreover, since the right side of the instances output by S are distributed similar to S', it follows that R together with S "inherit" the hardness of R' under S'.

Definition of S:
On input 1^n:

1. Choose i uniformly from $[0, n]$.
2. Choose a potential sampler $\langle M\rangle$ uniformly from $\{0,1\}^i$.
3. Run S' on 1^{n-i} (supplying it with random coins from the randomness tape of S itself) and denote the output by x.
4. Output $(\langle M\rangle, x)$.

We now formalize the ideas described above. We will use the following simple fact, that states that every potential sampler is output by S with noticeable probability:

Fact 3.2 *For every fixed potential sampler M_0, the probability that $S(1^n) = (\langle M_0\rangle, \cdot)$ is noticeable.*

Proof: By our definition, on input $1^{\ell+|\langle M_0\rangle|}$ the sampler S outputs a tuple $(\langle M\rangle, \cdot)$ where $|\langle M\rangle| = |\langle M_0\rangle|$ with probability $1/(\ell + |\langle M_0\rangle| + 1)$. Given that this event occurs, the probability that $\langle M\rangle = \langle M_0\rangle$ is fixed (namely, it is $2^{-|\langle M_0\rangle|}$). Thus, S outputs a tuple of the form $(\langle M_0\rangle, \cdot)$ with noticeable probability.

In the following we show that R and S are as required. Since R' is polynomially bounded, and λ is a fixed polynomial, it is straightforward that R too is polynomially bounded. It is easy to see that for some polynomial q (i.e., which depends on the model of computation) the relation R can be verified in time $q(\tau'(n))$ using one oracle call to a verifier for R'. Setting $\tau' = q^{-1}(\tau)$ we have that the relation R can be verified in time $\tau(n)$ using one oracle call to a verifier for R'. Since τ is super polynomial and time-constructible, it follows that $q^{-1}(\tau)$ is also super polynomial and time-constructible.

We show that R is hard under S. We first explain informally the idea of the proof. Let S_0 be some polynomial time onto sampler with randomness complexity bounded by λ. By Fact 3.2 any such S_0 is chosen by S with noticeable probability. As we will show, since R' is total, and since τ' is super-polynomial, for long enough x's, any instance of the form $(\langle S_0\rangle, x)$ is a YES-instance of R (and thus a solver should succeed on it). The second element in the pair, x, is chosen by S (independently from the first) according to the distribution of S'. It follows that conditioned on some noticeable event:

- The distribution on the x's is identical to the distribution of S'.
- The instances are always YES-instances.

Now, if a solver B does not fail on R under S (according to Definition 2.6), it does not fail on R under S conditioned on this event. This follows from (1) the fact that when this event occurs the instance is a YES-instance (2) the event is noticeable. Thus, failure on R under the distribution of S conditioned on this event implies failure on R under S (with no conditioning). Moreover, according to the definition of R, a solution for the instance $(\langle S_0\rangle, x)$ under R contains a solution for the instance x under R'. From all the above it follows that if B solves R under S, it implicitly solves in particular R' under S'. Thus the solver B can be used to solve R' under S', in contradiction.

We elaborate. Let S_0 be some polynomial time onto sampler with randomness complexity bounded by λ (for example, the sampler that on input 1^n outputs the first n coins from the randomness tape, which has randomness complexity bounded by λ since $\lambda(n) \geq n$). Let $p_{\tilde{S}_0}$ be a polynomial bounding the running time of $\tilde{S}_0$. By Fact 3.2, S outputs a tuple of the form $(\langle S_0\rangle, x)$ with noticeable probability. Let n_0 be such that for any $n \geq n_0$ it holds that $\tau'(n+|\langle S_0\rangle|) \geq p_{\tilde{S}_0}(n + |\langle S_0\rangle|)$ (clearly such number exists as τ' is super-polynomial).

Claim 3.3 *Any instance of the form $(\langle S_0\rangle, x)$ where $|x| \geq n_0$, is a YES-instance of R.*

Proof: The claim follows from the following facts:

- Every x has a w such that $(x, w) \in R'$, as R' is total.
- S_0 is an onto sampler, thus there are $r_1, \ldots, r_{|x|}$ such that for all $1 \leq i \leq |x|$, on input $1^{|\langle S_0\rangle|+|x|}$ and randomness r_i, the sampler $\tilde{S}_0$ outputs $(\langle S_0\rangle, x \oplus e_i^{|x|})$.
- Since the randomness complexity of S_0 is bounded by λ, it follows that for all $1 \leq i \leq |x|$, the randomness r_i can be taken such that $|r_i| \leq \lambda(|\langle S_0\rangle| + |x|)$.
- Since $|x| \geq n_0$ it follows that $p_{\tilde{S}_0}(|x| + |\langle S_0\rangle|) \leq \tau'(|x| + |\langle S_0\rangle|)$, thus S_0 halts on the r_i's in at most $\tau'(|\langle S_0\rangle| + |x|)$ steps as required.

Now, suppose to the contrary there exists an algorithm B that succeeds on R under S. That is,

$$\Pr_{(\langle M\rangle,x)\leftarrow S(1^n)}[((\langle M\rangle, x), B((\langle M\rangle, x))) \notin R \wedge \exists y((\langle M\rangle, x), y) \in R]$$

is not noticeable. Then (for $n \geq |\langle S_0\rangle|$):

$$\Pr_{(\langle M\rangle,x)\leftarrow S(1^n)}[((\langle M\rangle, x), B((\langle M\rangle, x))) \notin R \wedge \exists y((\langle M\rangle, x), y) \in R | \langle M\rangle = \langle S_0\rangle]$$

is also not noticeable, as we condition on a noticeable event (by Fact 3.2). Moreover, for large enough n's

$$\begin{aligned} &\Pr_{(\langle M\rangle,x)\leftarrow S(1^n)}[((\langle M\rangle, x), B((\langle M\rangle, x))) \notin R \\ &\qquad \wedge \exists y((\langle M\rangle, x), y) \in R | \langle M\rangle = \langle S_0\rangle] \\ = &\Pr_{(\langle M\rangle,x)\leftarrow S(1^n)}[((\langle M\rangle, x), B((\langle M\rangle, x))) \notin R | \\ &\quad \langle M\rangle = \langle S_0\rangle], \end{aligned}$$

as for large enough n's, when $\langle M\rangle = \langle S_0\rangle$ there is always a solution (by Claim 3.3). Since S chooses the second element (independent of the first) according to the distribution of S', it follows that for large enough n's, the last term equals

$$\Pr_{x\leftarrow S'(1^{n-|\langle S_0\rangle|})}[((\langle S_0\rangle, x), B((\langle S_0\rangle, x))) \notin R].$$

Let B_1 denote the algorithm that behaves similar to B but outputs only the first element (out of the pair) output by B. Then the last term, which by the above is not noticeable, upper bounds

$$\Pr_{x\leftarrow S'(1^{n-|\langle S_0\rangle|})}[(x, B_1((\langle S_0\rangle, x))) \notin R'],$$

since if B succeeds on the instance $(\langle S_0\rangle, x)$ then the first element output by B is a solution for x under R'.

Finally, trivially

$$\begin{aligned}&\Pr_{x\leftarrow S'(1^{n-|\langle S_0\rangle|})}[(x,B_1((\langle S_0\rangle,x)))\notin R']\\ \geq\ &\Pr_{x\leftarrow S'(1^{n-|\langle S_0\rangle|})}[(x,B_1((\langle S_0\rangle,x)))\notin R'\wedge\\ &\exists w(x,w)\in R']\end{aligned}$$

(in fact equality holds here as R' is total), and we conclude that the algorithm $B_1((\langle S_0\rangle,\cdot))$ does not fail with noticeable probability on R' under S', in contradiction to the hardness of R' under S'. We conclude that R is hard under S.

We now prove the main claim, i.e., that there does not exist a sampler S^* as above. Assume towards contradiction that a sampler S^* as above does exist. Then, we use (the code of) S^* to construct an algorithm A that succeeds (i.e., does not fail with noticeable probability) on R' under S', in contradiction to the assumed hardness of R' under S'. In fact, A will solve R' in the worst case, in expected polynomial time. We note, that this part of the proof only uses the fact that S^* is an onto sampler (which follows from the fact that the sampler S we construct is an onto sampler). Following is the definition of A. (See elucidating remarks below.)

Definition of A:
Let $p_{\tilde{S}^*}$ be a polynomial bounding the running time of $\tilde{S}^*$. Let n_0 be such that for any $n\geq n_0$ it holds that $\tau'(n)\geq p_{\tilde{S}^*}(n)$ (clearly such number exists as τ' is super-polynomial).

On input x of length n (an instance of R'):

```
1. If $n+|\langle S^*\rangle| < n_0$ output the answer out of a pre-computed (fixed) table.
2. Invoke $S^*(1^{n+|\langle S^*\rangle|})$ (supplying it with random coins). Denote the output by $((\langle M\rangle,x^{(0)}),y)$.
3. If $\langle M\rangle\neq\langle S^*\rangle$ go back to 2. Else, If $\langle M\rangle=\langle S^*\rangle$, proceed (note that in that case $|x^{(0)}|=|x|=n$).
4. Parse $y$ to $(w^{(0)},r_1^{(0)},\ldots,r_n^{(0)})$ (the fact that $y$ can be so parsed will be proven later).
5. Let $i_1,\ldots,i_h$ be the locations of the bits that are different between $x$ and $x^{(0)}$.
   For $j=1$ to $h$:
       Use $r_{i_j}^{(j-1)}$ as randomness for $S^*(1^{n+|\langle S^*\rangle|})$ to
   obtain output $((\langle M\rangle,x^{(j)}),(w^{(j)},r_1^{(j)},\ldots,r_n^{(j)}))$
       (Again, the fact that the second element in the output of $S^*$ can be so parsed will be proven later. Also, we will show that it must hold that $\langle M\rangle=\langle S^*\rangle$ and $x^{(j)}=x^{(j-1)}\oplus e_{i_j}^n$.)
6. Output $w^{(h)}$.
```

Before proving the correctness and the running time of the algorithm, let us explain informally its idea. In Lines 2 and 3 the algorithm invokes S^* (with appropriate input-length) until it outputs an instance-solution pair where the instance is of the form $(\langle S^*\rangle,x^{(0)})$ (where, due to the choice of the input-length, the length of $x^{(0)}$ is $|x|$). As we will show, this part takes expected polynomial time. The idea is that once such an instance is output, the string y that accompanies it is a solution with respect to R (since the instance is a YES-instance and S^* is assumed to output solutions when they exist). Moreover, this solution relates to the machine S^* *itself.* That is, y is of the form $(w^{(0)},r_1^{(0)},\ldots,r_n^{(0)})$, where the $r_i^{(0)}$'s can be used as randomness for S^* to "flip" the i-th bit of $x^{(0)}$, and obtain a new pair $((\langle S^*\rangle,x^{(1)}),(w^{(1)},r_1^{(1)},\ldots,r_n^{(1)}))$, where $x^{(1)}$ is Hamming-closer to x by one bit. Then, the same idea is repeated in the loop in Line 5. At each step j the algorithm chooses the j-th bit that is different between $x^{(0)}$ and x, and using the appropriate randomness from the previous solution (i.e., $r_{i_j}^{(j-1)}$) "flips" it to become Hamming-closer to x by one bit. After reaching x (that is, after having S^* output a pair where the instance is $(\langle S^*\rangle,x)$), in Line 6, the algorithm outputs $w^{(h)}$, which, by the definition of R, and since S^* always outputs legal solutions with respect to R when these exist, is a solution for x under R'.

We proceed to a formal proof.

Claim 3.4 *The algorithm A runs in expected polynomial time.*

Proof: We first show that the algorithm reaches Line 4 in expected polynomial time. By Fact 3.2 the probability of the event that on input $1^{n+|\langle S^*\rangle|}$ the sampler S outputs $(\langle S^*\rangle,x^{(0)})$ (for some $x^{(0)}$ with $|x^{(0)}|=n$) is noticeable in n.

Since by assumption $\tilde{S}^*$ dominates S, it follows that the event that on input $1^{n+|\langle S^*\rangle|}$ the sampler S^* outputs $(\langle S^*\rangle,x^{(0)})$ (for some $x^{(0)}$ with $|x^{(0)}|=n=|x|$) is also noticeable in n. It follows that the algorithm reaches Line 4 in expected polynomial time. It is easy to see (considering that the running time of S^* is polynomial) that Lines 5,6 run in (strict) polynomial time. The claim follows.

We now prove the correctness of the algorithm. That is, we show that the output of the algorithm A on input x is always a solution for x under R'.

Claim 3.5 *If the algorithm A reaches Line 4, then y is a valid solution for $(\langle M\rangle,x^{(0)})$ under R.*

Proof: We show that when the algorithm reaches Line 4, $(\langle M\rangle,x^{(0)})$ is a YES-instance, and therefore y is a valid solution for $(\langle M\rangle,x^{(0)})$ (as by assumption the second element in every pair output by S^* is a solution for the first element with respect to R if such exists).

The proof resembles the proof of Claim 3.3. Note that if the algorithm reaches Line 4 then $\langle M \rangle = \langle S^* \rangle$. We then note that:

- $x^{(0)}$ has a w such that $(x^{(0)}, w) \in R'$, as R' is total.
- $\tilde{S^*}$ is an onto sampler since S is an onto sampler, and $\tilde{S^*}$ dominates S. Thus, there are $r_1^{(0)}, \ldots, r_{|x|}^{(0)}$ such that for all $1 \le i \le |x|$, on input $1^{|\langle S^* \rangle|+n}$ and randomness $r_i^{(0)}$, the sampler $\tilde{S^*}$ outputs $(\langle S^* \rangle, x \oplus e_i^{|x|})$.
- Since the randomness complexity of S^* is assumed to be bounded by λ, it follows that for all $1 \le i \le |x|$, the randomness $r_i^{(0)}$ can be taken such that $|r_i^{(0)}| \le \lambda(|\langle S^* \rangle| + n)$.
- If A reaches Line 4 then $n + |\langle S^* \rangle| \ge n_0$ (because of Line 1), thus $\tau'(n + |\langle S^* \rangle|) \ge p_{\tilde{S^*}}(n + |\langle S^* \rangle|)$. Thus $\tilde{S^*}$ halts on the $r_i^{(0)}$'s in at most $\tau'(|\langle S^* \rangle| + n)$ steps as required.

The claim follows.

Finally, we show that the algorithm outputs the correct output. First we note:

Claim 3.6 *The second element in the output of S^* in each step j in the loop in Line 5 is a valid solution for $(\langle M \rangle, x^{(j)})$ with respect to R.*

The proof is identical to the proof of Claim 3.5.

Claim 3.7 *For every j, at the end of step j in the loop in Line 5 the Hamming distance between $x^{(j)}$ and x is $h - j$.*

Proof: By Claims 3.5 and 3.6, and since by assumption the second element in every pair output by S^* is a solution for the first element with respect to R if such exists, it follows that throughout the algorithm, "all $r_i^{(j)}$'s are correct" (formally, for every $0 \le j \le h$ and every $1 \le i \le n$, any $r_i^{(j)}$ is such that $S^*(1^{n+|\langle S^* \rangle|})$ with randomness $r_i^{(j)}$ outputs a pair of the form $((\langle S^* \rangle, x^{(j)} \oplus e_i^n), \cdot))$. Since in step j the algorithm A chooses to run S^* on $r_{i_j}^{(j-1)}$, where i_j is the j-th bit that is different between $x^{(0)}$ and x, it follows that $x^{(j+1)}$ is Hamming-closer to x than $x^{(j)}$ by one bit. The claim follows.

It follows that $x^{(h)} = x$. Since $(w^{(h)}, r_1^{(h)}, \ldots, r_n^{(h)})$ is a solution for $(\langle M \rangle, x^{(h)})$ under R (again, by Claim 3.6), by the definition of R it follows that $w^{(h)}$ is a solution for $x^{(h)} = x$ under R'. The theorem follows.

4 Conclusion

4.1 On Our Assumptions

We have shown that if there exists a polynomially bounded total search problem that is hard under some polynomial time onto sampler, then there exist efficiently samplable distributions that are hard on average, but that cannot be transformed into one-way functions (in the manner we described).

We note that the hypothesis of the theorem is implied by the existence of one-way permutations, is further implied by the weaker assumption of the existence of onto one-way functions (where by "onto" we mean that the function's image is Σ^*), and is even weaker than the latter. Thus, our assumptions are plausible. In fact, one does not even have to assume that S' is an onto sampler. If S' is a sampler with the property that size of its image is noticeable, it can be made onto by standard (straightforward) techniques without harming the hardness of any search problem under this sampler.

It might seem contradictory at first sight that we mention that our hypothesis is implied by an assumption that in fact yields the *existence* of one-way functions. But, our result is essentially about approaches to *construct* one-way functions, and not about their *existence.* Thus, a way to interpret the result, is that if onto one-way functions exist, then a certain way to prove the existence of one-way functions cannot work (although one-way functions in fact do exist under this assumption).

4.2 On Restricting the Randomness Complexity of S^*

To the best of our knowledge, it is unknown if restricting the randomness of the pairs-sampler S^* is with loss of generality. It is fairly easy to see that if the answer to the following (open, to the best of our knowledge) question is affirmative, then restricting the randomness of the pairs-sampler S^* is *without* loss of generality.

Open Question 4.1 *There exists a (universal) polynomial q such that for every pairs-sampler S^* such that $\tilde{S^*}$ is an onto sampler, there exists a pairs-sampler S^{**} such that $\tilde{S^{**}}$ is an onto sampler and such that:*

- *For every n, it holds that* $\text{support}(S^{**}(1^n)) \subseteq \text{support}(S^*(1^n))$.
- *The sampler $\tilde{S^{**}}$ dominates $\tilde{S^*}$.*
- *The randomness complexity of S^{**} is bounded by q.*

4.3 Possible Interpretations of Our Result

First we note that the conclusion of our theorem can be presented in a different way: *For any polynomials q and λ there exists a polynomial time verifiable search problem R and a polynomial time sampler S such that R is hard under S, but where any sampler S^* as in the Desired Construction must run in time exceeding q and use randomness exceeding λ.* It is easy to modify the proof accordingly (basically, one just has to change the function τ in the definition of R, to q).

A possible interpretation of our result is as an indication for the generality of the classes for which one can achieve the Desired Construction. Following the discussion in the Introduction, the most generic manner to achieve the Desired Construction is to achieve it for any R and S where R is hard under S, and where R is polynomially bounded and exponential time verifiable. Under the restriction on the randomness complexity of S^*, we eliminate the possibility to achieve this generic construction.

We note, however, that our result does not imply that any "generic" approach to achieve the Desired Construction cannot succeed (where by "generic" we mean a transformation that works for a wide class of pairs (R, S)). The two main reasons for this statement (besides the loss of generality implied by the restriction on the randomness complexity of S^*) are:

- The search problem we construct is not polynomial time verifiable (but is "nearly polynomial time verifiable"). This still leaves open the possibility that any polynomial time verifiable search problem that is hard on average under some efficient sampler can be transformed to a one-way function (via the Desired Construction we refer to).
- We require that the desired S^* relates strictly to R and S. A slightly different approach could be to first transform R and S to some R_1 and S_1, and then construct S^* for R_1 and S_1. Our result does not eliminate the possibility that this approach can work for a wide class of pairs (R, S).

Acknowledgments

I would like to deeply thank Oded Goldreich for many helpful discussions, for the help in the presentation of this paper, and for encouraging me to work in this area.

References

[1] A. Akavia, O. Goldreich, S. Goldwasser, and D. Moshkovitz. On basing one-way functions on np-hardness. In J. M. Kleinberg, editor, *STOC*, pages 701–710. ACM, 2006.

[2] A. Bogdanov and L. Trevisan. On worst-case to average-case reductions for np problems. *SIAM J. Comput.*, 36(4):1119–1159, 2006.

[3] G. Brassard. Relativized cryptography. In *FOCS*, pages 383–391. IEEE, 1979.

[4] J. Feigenbaum and L. Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993.

[5] O. Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.