

# Exposure-resilient extractors

Marius Zimand \*

March 10, 2006

## Abstract

An exposure-resilient extractor is an efficient procedure that, from a random variable with imperfect min-entropy, produces randomness that passes all statistical tests including those that have bounded access to the random variable, with adaptive queries that can depend on the string being tested. More precisely,  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  is a  $(k, \epsilon)$ -exposure resilient extractor resistant to  $q$  queries if, when the min-entropy of  $x$  is at least  $k$  and  $y$  is random,  $\text{EXT}(x, y)$  looks  $\epsilon$ -random to all statistical tests modeled by oracle circuits of unbounded complexity that can query  $q$  bits of  $x$ . We construct, for any  $\delta < 1$ , a  $(k, \epsilon)$ -exposure resilient extractor with query resistance  $n^\delta$ ,  $k = n - n^{\Omega(1)}$ ,  $\epsilon = n^{-\Omega(1)}$ ,  $m = n^{\Omega(1)}$  and  $d = O(\log n)$ .

## 1 Introduction

Extractors are efficient procedures for improving the quality of a source of randomness. A well-studied type is the *seeded extractor*, that takes as input two strings. The first one is called the *weakly-random string* and its salient feature is that its distribution (among the strings of a given length  $n$ ) is arbitrary except for the fact that its min-entropy is guaranteed to be larger than some parameter. The second input is called the *seed*, it is much shorter than the weakly random string (usually the length is  $\log n$  or  $\text{polylog } n$ ), and is uniformly distributed. The output of the extractor is required to be at a small statistical distance from the uniform distribution. Thus, roughly speaking, an extractor is an efficient procedure that converts imperfect randomness (where the degree of imperfectness is measured by the min-entropy of the weakly-random string) into randomness that is statistically close to perfect. Seeded extractors have numerous applications (see the surveys by Nisan [Nis96], Vadhan [Vad], Shaltiel [Sha02]), including cryptography. The concept of statistical distance is based on the concept of a statistical test, and, in a cryptography context, it is appropriate to view a statistical test as an adversary that wants to distinguish between the extractor's output, denoted  $\text{EXT}(x, y)$  (where  $x$  is the weakly-random string and  $y$  is the seed), and the uniform distribution. An important theme of research in cryptography is the investigation of situations that deviate in one sense or another from the standard assumptions (e.g., the situation in which the adversary has some information regarding a secret key). Generically, this is called the study of *exposure resiliency* of various crypto primitives or protocols [Dod00]. Following this line of research, in this paper we study extractors that are stronger than standard extractors in one interesting aspect. Namely, we require that the extractor's output appears to be random even to statistical tests that can query the weakly-random string a bounded number of times. Other than the number of bits that can be queried, there is no other restriction, i.e., the adversary is not computationally bounded in any way and the queries can be adaptive (each query can depend on the answers to the previous queries). There already have been studies in which the

---

\*Department of Computer and Information Sciences, Towson University, Baltimore, MD.  
<http://triton.towson.edu/~mzimand>

adversary that attempts to distinguish between  $\text{EXT}(x, y)$  and the uniform distribution is allowed to have some extra information. For example, in the case of *strong extractors*, the adversary knows the seed. Lu [Lu04] and later Vadhan [Vad04], motivated by the utilization of extractors in the bounded-storage-model cryptography, have considered the case in which the adversary works in two rounds; in the first round the adversary accesses the weakly-random string and stores  $q$  bits of information, and in the second round the adversary is given access to  $\text{EXT}(x, y)$  and to the seed  $y$ . Note that for such extractors, that will be referred here as *Lu extractors*, the queries to  $x$  do not depend on  $\text{EXT}(x, y)$ . In contrast, we consider the case in which the queries depend on  $\text{EXT}(x, y)$  and are adaptive. Lu [Lu04] has shown that basically a strong extractor is a Lu extractor (with slightly weaker parameters).

Thus the distinction between exposure-resilient extractors (introduced in this paper) and other types of seeded extractors consists in (1) the extra information that the adversary has and (2) the manner and the timing at which the adversary can obtain this information. We list below in a more formal way all these types of extractors emphasizing the differentiating features. A seeded extractor is a polynomially-computable function  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ , in which the first input,  $x$ , has min-entropy  $H_\infty(x) \geq k$  and the second output,  $y$ , is uniformly distributed. Actually for the polynomial time complexity condition to make sense, implicitly we have in mind a uniform family of such functions, one for each  $n$ , and with the other parameters being functions of  $n$ .  $U_m$  denotes the uniform distribution on  $\{0, 1\}^m$ . By adversary we mean a circuit of arbitrary size.

- (a) (*Standard extractor*) The adversary is given the challenge  $Z$  which is either (1)  $\text{EXT}(x, y)$ , or (2)  $U_m$ . The adversary wants to distinguish between the cases (1) and (2) with bias  $\epsilon$ . The function  $\text{EXT}$  is a  $(k, \epsilon)$  extractor if there is no adversary that succeeds.
- (b) (*Strong extractor*) The adversary is given the seed  $y$  and the challenge  $Z$  which is either (1)  $\text{EXT}(x, y)$ , or (2)  $U_m$ . The adversary wants to distinguish between the cases (1) and (2) with bias  $\epsilon$ . The function  $\text{EXT}$  is a  $(k, \epsilon)$  strong extractor if there is no adversary that succeeds.
- (c) (*Lu extractor*) The adversary first receives  $x$  and is allowed to calculate  $f(x)$  for some function  $f$ , so that  $|f(x)| = q < n$ . Then the adversary loses access to  $x$  and is given the seed  $y$  and the challenge  $Z$  which is either (1)  $\text{EXT}(x, y)$ , or (2)  $U_m$ . The adversary wants to distinguish between the cases (1) and (2) with bias  $\epsilon$ . The function  $\text{EXT}$  is a  $(k, \epsilon)$  Lu extractor if there is no adversary that succeeds.
- (d) (*Exposure-resilient extractor - ERE*) The adversary is given the challenge  $Z$  which is either (1)  $\text{EXT}(x, y)$ , or (2)  $U_m$ , and *simultaneously* oracle access to  $x$  to which it is allowed to make  $q$  queries. The adversary wants to distinguish between the cases (1) and (2) with bias  $\epsilon$ . The function  $\text{EXT}$  is a  $(k, \epsilon)$  exposure-resilient extractor resistant to  $q$  queries if there is no adversary that succeeds.

A related notion is that of an exposure-resilient function (ERF). The strongest variant (and closest to EREs) is as follows. Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ . The input  $x \in \{0, 1\}^n$  is chosen uniformly at random. The adversary is given the challenge  $Z$  which is either (1)  $f(x)$ , or (2)  $U_m$ , and can query adaptively  $(n - q)$  bits of  $x$ . The adversary wants to distinguish between the cases (1) and (2) with bias  $\epsilon$ . The function  $f$  is a  $q$ -ERF if there is no adversary that succeeds.

EREs are in a sense less general than ERFs because the input consists of two parts - the weakly-random string and the seed, and the adversary is only given access to the weakly-random string. Thus some (small) fixed part of the input is considered to be completely hidden from the adversary.

In another aspect, however, EREs are more general than ERFs because in an ERF the input is completely random (i.e., it is uniformly distributed in  $\{0, 1\}^n$ ) while in an ERE a large part of the input, namely the weakly-random string, is allowed to have min-entropy smaller than its length.

The fact that in an ERF the adversary can query  $x$  in an adaptive manner creates difficulties in the analysis. To circumvent this problem, Dodis, Sahai and Smith [DSS01] consider the notion of almost-perfect resilient functions (APRF). A function  $f(x)$  is an APRF if for any set  $L$  of  $n - q$  positions in the string  $x \in \{0, 1\}^n$  and for any assignment  $w$  to those positions in  $x$ ,  $f(x)$  is almost random when the remaining  $q$  bits of  $x$  are random. More formally, let  $[x]_L$  be the projection of  $x$  on the positions in  $L$ . The function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  is a  $q$ -APRF if for any set  $L$  of  $n - q$  positions, for all  $w$  and  $y$ ,

$$|\text{Prob}[f(x) = y \mid [x]_L = w] - 2^{-m}| < \epsilon(n),$$

where  $\epsilon(n)$  is a negligible function. APRFs are useful because on one hand their definition is purely combinatorial (no adversary is involved) and, on the other hand, Dodis, Sahai and Smith [DSS01] have shown that a  $q$ -APRF is also a  $q$ -ERF. The paper [DSS01] gives a probabilistic construction of an  $\ell$ -ERF function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ , for  $\ell = \omega(\log n)$  and  $m = \ell - o(\log \ell)$ . Kamp and Zuckerman [KZ03], using the connection with APRFs as well, give an explicit construction of an  $n^{1/2+\gamma}$ -ERF function  $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$  with  $m = n^{2\gamma}$ .

**OUR RESULT.** One contribution of this work is at the conceptual level. Exposure-resilient extractors are based on a new type of distance between distributions and thus they are fundamentally different from all previously-studied extractors. The definitions of all extractors in the literature state their main requirement in terms of *statistical distance*. At its turn, this distance is defined in terms of a type of test that has a *passive* nature and that can be viewed as a fixed target that the extractor has to hit in the right way. Indeed, a test is simply a set  $A$  and two distributions  $X$  and  $Y$  are  $\epsilon$ -distinguishable by  $A$  if  $|\text{Prob}(X \in A) - \text{Prob}(Y \in A)| > \epsilon$ . The distributions  $X$  and  $Y$  are  $\epsilon$ -statistically close if there is no test that  $\epsilon$ -distinguishes them. In order to foil such passive tests, the extractor's output has to satisfy certain combinatorial properties and, consequently, (standard) extractors can be obtained using techniques similar to those used to build  $k$ -wise independent random variables or error-correcting codes. The same type of techniques have been employed for building APFRs. On the other hand, exposure-resilient extractors are defined in terms of a type of test that has an *active*, algorithmical nature, and that corresponds to a target that can move to a certain degree trying, say, to escape the extractor's hits (see Definition 2.1). We consider that such tests define a type of distance that is natural and mathematically interesting in its own. Moreover, they model a stronger type of adversary and, consequently, have the potential of being useful in cryptography.<sup>1</sup> They can also be used to sample a set that has some dependency on the randomness of the sampler. Because of the algorithmical nature of the tests that exposure-resilient extractors have to foil, it seems that combinatorial techniques are no longer sufficient for building such extractors and one has to recourse to computational-complexity methods (we elaborate more the discussion on techniques in Section 3).

The other, more concrete, contribution of this paper is the construction of a polynomial-time computable  $(k, \epsilon)$  exposure-resilient extractor  $\text{EXT} : \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ , resistant to  $n^\delta$  queries, where  $\delta$  is arbitrarily close to 1. The min-entropy is  $k = n - n^{\Omega(1)}$ , the bias  $\epsilon = n^{-\Omega(1)}$  and the output length is  $m = n^{\Omega(1)}$ . The seed length is  $d = O(\log n)$ . The algorithm that computes

---

<sup>1</sup>For instance, (standard) extractors can counter an adversary that has the capability of perturbing a high-quality randomness source in a way that reduces its min-entropy (this application of extractors is considered in [BST03]). An exposure-resilient extractor can foil adversaries that in addition to the above have direct bounded access to the weakly-random string (as in crypto scenarios in which ERFs are useful).

the extractor is simple and efficient and reads fewer bits from the weakly-random string than the adversary.

Compared to ERFs, the query resistance is low and, compared to standard seeded extractors, the min-entropy is high and the output length is small. However, the result is not trivial. As we have argued above, exposure-resilient extractors are in a qualitative way different from both standard seeded extractors and ERFs and they require a different type of techniques. Nevertheless, we have no reason to believe that the above parameters are optimal. Their improvement is left as an open problem.

## 2 Definitions

Notations:  $x \odot y$  denotes the concatenation of the strings  $x$  and  $y$ ,  $|x|$  denotes the length of the string  $x$ , and  $\|A\|$  denotes the cardinality of the set  $A$ . Let  $n \in \mathbf{N}$ . We assume throughout this paper that the length  $\tilde{n}$  of the weakly random string  $X$  is of the form  $\tilde{n} = n2^n$ , for some  $n \in \mathbf{N}$ . We view  $X$  as the truth-table of a function mapping  $\{0, 1\}^n$  to  $\{0, 1\}^{\tilde{n}}$  and we call this function  $X$  as well. The min-entropy of a random variable  $X$  taking values in  $\{0, 1\}^{\tilde{n}}$  is given by  $\min \left\{ \log \frac{1}{\text{Prob}(X=a)} \mid a \in \{0, 1\}^{\tilde{n}}, \text{Prob}(X = a) \neq 0 \right\}$ . The min-entropy of  $X$  is denoted  $H_\infty(X)$ . Thus if  $X$  has min-entropy  $\geq k$ , then for all  $a$ ,  $\text{Prob}(X = a) \leq 1/2^k$ . For each  $n \in \mathbf{N}$ , let  $U_n$  denote the uniform distribution over  $\{0, 1\}^{\tilde{n}}$ .

**Definition 2.1** (Adaptive Test) *An adaptive test is an oracle circuit that uses a functional oracle  $X$  of the type  $X : \{0, 1\}^n \rightarrow \{0, 1\}^{\tilde{n}}$ . A query to  $X$  is a string  $z \in \{0, 1\}^n$  and the oracle answers with  $X(z)$ . We say that the adaptive test has query complexity  $Q$  if the oracle circuit is allowed to make at most  $Q$  queries.*

If  $D$  is an adaptive test, we denote by  $D^X$  the set of strings accepted by  $D$  when using the oracle  $X$ .

**Definition 2.2** (Exposure Resilient Extractor) *The values  $\tilde{n}, k, d, m$ , and  $Q$  are integer parameters, and  $\epsilon > 0$  is a real number parameter. A function  $E : \{0, 1\}^{\tilde{n}} \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  is a  $(k, \epsilon)$ -exposure resilient extractor resistant to  $Q$  queries if for every distribution  $X$  on  $\{0, 1\}^{\tilde{n}}$  with min-entropy at least  $k$  and for every adaptive test  $D$  with query complexity  $Q$ ,*

$$|\text{Prob}_{X, y \in \{0, 1\}^d}(E(X, y) \in D^X) - \text{Prob}_{X, z \in \Sigma^m}(z \in D^X)| \leq \epsilon. \quad (1)$$

As in the case of standard extractors, we implicitly have in mind a family of extractors indexed by  $n$  and with the parameters  $k, d, m, Q$ , and  $\epsilon$  being functions of  $n$ . We also want the extractor to run in polynomial time. If  $D$  is an adaptive test, we say that  $X$  hits  $D$   $\epsilon$ -correctly if

$$\left| \frac{\|\{E(X, y) \mid y \in \{0, 1\}^d\} \cap D^X\|}{\|\{0, 1\}^d\|} - \frac{\|D^X\|}{\|\{0, 1\}^m\|} \right| \leq \epsilon.$$

The following is an analog of a basic property of standard seeded extractors.

**Lemma 2.3** *Let  $E : \{0, 1\}^{\tilde{n}} \times \{0, 1\}^d \rightarrow \{0, 1\}^m$  and  $\epsilon > 0$ . Suppose that for every adaptive test  $D$  with query complexity  $Q$  the number of  $x \in \{0, 1\}^{\tilde{n}}$  that do not hit  $D$   $\epsilon$ -correctly is at most  $2^t$ , for some  $t$ . Then  $E$  is a  $(t + \log(1/\epsilon), 2\epsilon)$ -extractor that is resistant to  $Q$  queries.*

**Proof** Let  $X$  be a distribution on  $\{0, 1\}^{\tilde{n}}$  with min-entropy  $t + \log(1/\epsilon)$ . Let  $D$  be an adaptive test with query complexity  $Q$  on inputs of length  $m$ . Consider the set  $B = \{x \in \{0, 1\}^{\tilde{n}} \mid$

$x$  does not hit  $D$   $\epsilon$ -correctly}. This set has at most  $2^t$  elements and the distribution  $X$  allocates to this set a probability mass of at most  $2^t \cdot 2^{-(t+\log(1/\epsilon))} = \epsilon$ . We have:

$$\begin{aligned} \text{Prob}_{X,y}[E(X,y) \in D^X] &= \\ &\text{Prob}_{X,y}[E(X,y) \in D^X \text{ and } X \text{ hits } D^X \text{ } \epsilon\text{-correctly}] \\ &+ \text{Prob}_{X,y}[E(X,y) \in D^X \text{ and } X \text{ does not hit } D^X \text{ } \epsilon\text{-correctly}] \end{aligned}$$

The second term is bounded by the probability that  $X$  does not hit  $D$   $\epsilon$ -correctly and thus is in the interval  $[0, \epsilon]$ . We claim that the first term is in the interval  $[\text{Prob}_{X,z}[z \in D^X] - 2\epsilon, \text{Prob}_{X,z}[z \in D^X] + \epsilon]$ . The conclusion follows immediately from these two facts. We now prove the claim. The first term is equal to  $\sum_{u \in \bar{B}} \text{Prob}_X[X = u] \cdot \text{Prob}_y[E(u,y) \in D^u]$ , which is at most

$$\begin{aligned} \sum_{u \in \bar{B}} \text{Prob}[X = u] \cdot \left( \frac{\|D^u\|}{2^m} + \epsilon \right) \\ \leq \sum_{u \in \{0,1\}^{\bar{n}}} \text{Prob}[X = u] \cdot \frac{\|D^u\|}{2^m} + \epsilon \cdot \sum_{u \in \{0,1\}^{\bar{n}}} \text{Prob}_X[X = u] \\ = \text{Prob}_{X,z}[z \in D^X] + \epsilon. \end{aligned}$$

On the other hand, the first term is at least

$$\begin{aligned} \sum_{u \in \bar{B}} \text{Prob}[X = u] \cdot \left( \frac{\|D^u\|}{2^m} - \epsilon \right) \\ \geq -\epsilon + \sum_{u \in \bar{B}} \text{Prob}[X = u] \cdot \frac{\|D^u\|}{2^m} \\ = -\epsilon + \sum_{u \in \{0,1\}^{\bar{n}}} \text{Prob}[X = u] \cdot \frac{\|D^u\|}{2^m} - \sum_{u \in B} \text{Prob}[X = u] \cdot \frac{\|D^u\|}{2^m} \\ \geq -2\epsilon + \text{Prob}_{X,z}[z \in D^X]. \end{aligned}$$

■

### 3 Overview of proof

It is difficult to analyze cryptographic primitives and protocols in which the adversary has fully adaptive attack capabilities and such results are rare. In the Introduction, we have recalled the case of exposure-resilient functions (ERFs) for which such an analysis has been accomplished via a reduction to another primitive, namely the almost-perfect resilient function (APRF), that does not involve in its definition an adaptive adversary. This reduction does not seem to extend to the case in which the adversary has access to some information whose min-entropy is smaller (even by 1 bit) than its bit-length, which is exactly the case of exposure-resilient extractors. Instead, we are using a reduction to one-way length-preserving functions, a primitive for which adaptive adversaries can be analyzed. Indeed, there are general results due to Impagliazzo [Imp96], Gennaro and Trevisan [GT00], Wee [Wee05], showing that a random function  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is, with high probability, one-way for adversaries that can pose a bounded number of queries to the truth-table of  $R$ , even if the queries are adaptive. Using the proof technique from [Imp96], we prove in Section 4 such a result for a combination of parameters that is appropriate for EREs. It also holds that random length-preserving functions do not have any string in the range with a large preimage set. It is known from the work of Hastad, Impagliazzo, Levin and Luby [HILL99] how to use a one-way function with the "no-large-preimage-set" property to construct a pseudo-random generator. However, we need to construct extractors and not pseudo-random generators. Pseudo-random

generators are the counterparts of extractors in the computational setting, i.e., while the output of an extractor is close to uniform in the statistical sense, the output of a pseudo-random generator is close to uniform only in the computational sense. Despite this drastic difference, very surprisingly, Trevisan [Tre01] has shown that some techniques for constructing pseudo-random generators can be used to produce extractors. Specifically, Trevisan [Tre01] shows that the Nisan-Wigderson construction of a pseudo-random generator from a hard function can be used almost directly to build extractors (the extra needed ingredient is an error-correcting code with good list-decoding properties). Recently, Zimand [Zim05] has shown that the other main avenue for building pseudo-random generators, the Blum-Micali-Yao construction of a pseudo-random generator (using as a building block a one-way length-preserving permutation) also yields extractors. The techniques from either [Tre01] or [Zim05] do not yield exposure-resilient extractors, but, fortunately, the construction from [HILL99] of a pseudo-random generator that uses as a building block a one-way function  $R$  with small preimage sets does produce exposure-resilient extractors. This construction uses  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$  as a black-box and constructs a function  $g_R$  that stretches the input (i.e.,  $|g_R(x)| \gg |x|$ ) and which has the following property. If there exists a circuit  $D$  that distinguishes  $g_R(x)$  from a random string, when  $x$  is uniformly at random chosen in the domain of  $g_R$ , then there is a circuit  $A$  not much larger than  $D$  that inverts  $R$  on many inputs. Therefore, if  $R$  is one-way there is no distinguisher circuit  $D$ . In the style of [Tre01], we view the truth-table of  $R$  as the first input (the weakly-random string) of the extractor that we build. The fact that  $D$  distinguishes  $g_R$  from the uniform distribution with bias  $\epsilon$  is equivalent to saying  $R$  does not hit  $D$   $\epsilon$ -correctly. We can allow here  $D$  to query  $R$  adaptively. Then from the analysis we deduce that there is a circuit  $A$  making only a small number of additional queries to  $R$  that inverts  $R$  on many inputs. As mentioned above (and as proven in Section 4), for any given oracle circuit  $A$ , there are only few  $R$ 's with the above property. Therefore only a few  $R$ 's do not hit  $D$   $\epsilon$ -correctly and thus, according to Lemma 2.3, we have obtained an extractor. The actual proof is a little bit more complicated because the circuit  $A$  is probabilistic and we cannot embed into  $A$  the "best" random bits (because the "best" random bits may depend on  $R$ ). Therefore we obtain several circuits  $A$  (one for each fixing of the random bits) and we know that one of these circuits inverts  $R$  on many inputs while making a bounded number of queries to  $R$ . This still allows us to bound the number of  $R$ 's that do not hit  $D$   $\epsilon$ -correctly.

## 4 Random one-way functions

A random function is with high-probability one-way (for algorithms that have bounded oracle access to it). Variants of this fact have been shown for different combinations of parameters by Impagliazzo [Imp96], by Gennaro and Trevisan [GT00], and by Wee [Wee05]. The next lemma is based on the techniques of Impagliazzo [Imp96] and allows for a flexible choice of parameters. It establishes that most functions  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$  do not have any element in the range with a large pre-image set and that any circuit with bounded access to a random such  $R$  can not invert but a small subset of the range of  $R$ . We fix a length  $n$ , and we consider a random function  $R : \Sigma^n \rightarrow \Sigma^n$ . We use the notation  $N = 2^n$ . For integers  $n$  and  $k$ , the product  $n(n-1) \dots (n-k+1)$  is denoted as  $(n)_k$ ; an  $n$ -set is a set with cardinality  $n$ ;  $e$  is the base of the natural logarithm. Let  $S$  and  $T$  be two integer parameters so that  $ST \leq N$ , let  $\alpha \in (0, 1)$  with  $\log(1/\alpha) < n$ . The reader should think of the case  $\alpha ST \ll 1$ . Let  $C$  be a probabilistic oracle circuit that queries the oracle at most  $S$  times for all oracles and all probabilistic branches. The oracle is a function such as  $R$ , and on a query  $q$ , where  $q$  is an element in the domain of  $R$ , it returns  $R(q)$ . We write  $C^R$  to mean that the circuit  $C$  uses the function  $R$  as the oracle. The notation  $C^R(y, \rho)$  represents the computation done by  $C^R$  on input  $y$  and using the random bits provided by the binary string  $\rho$ .

**Lemma 4.1** (a) Let  $E$  be the event (over random  $R$ ) “ $R$  is not  $\sqrt{\alpha N}$ -to-1” The probability of  $E$  is bounded by  $2^{-\Omega(n\sqrt{\alpha N})}$ .

(b) Let  $B$  be the event (over random pairs  $(R, \rho)$ ) “ $\|\{x \in I \mid C^R(R(x), \rho) \in R^{-1}(R(x))\}\| \geq 2e \cdot \alpha N \cdot S \cdot T$ ”. The probability of  $B$  is bounded by  $2^{-T} + 2^{-\Omega(n\sqrt{\alpha N})}$

(c) We say that  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is “good” for a probabilistic oracle circuit  $C$  if with probability of  $\rho$  greater than  $1/2$  it holds that

$$\|\{x \in \{0, 1\}^n \mid C^R(R(x), \rho) \in R^{-1}(R(x))\}\| \geq 2e \cdot \alpha N \cdot S \cdot T.$$

Let  $G$  be the event (over random  $R$ ) “ $R$  is good for  $C$ .” Then the probability of  $G$  is at most  $2 \cdot (2^{-T} + 2^{-\Omega(n\sqrt{\alpha N})})$ .

**Proof** (a) Let  $P = \sqrt{\alpha N}$ . Fix  $y \in \{0, 1\}^n$ , and a  $P$ -subset  $\{x_1, \dots, x_P\} \subset \{0, 1\}^n$ . The probability that all the elements of the  $P$ -set map into  $y$  is  $(1/N)^P$ . Therefore the probability that there exists  $y \in \{0, 1\}^n$  and  $P$  elements of  $\{0, 1\}^n$  mapping into  $y$  is at most  $N \cdot \binom{N}{P} \cdot (1/N)^P < N(eN/P)^P (1/N)^P = N(e/P)^P = 2^{-P(\log P - \log e) + n} = 2^{-\sqrt{\alpha N}((1/2)(n - \log(1/\alpha)) - \log e) + n}$ .

(b) We show that  $\text{Prob}(B \mid \bar{E}) \leq 2^{-T}$ , where  $\bar{E}$  is the complement of  $E$ . From here point (b) follows because  $\text{Prob}(B) \leq \text{Prob}(B \mid \bar{E}) + \text{Prob}(E)$ . We can assume that the circuit  $C$  on input  $y = R(x)$  queries the oracle  $R$  about its output  $z$  to check whether  $R(z) = y$ . Let  $C'$  be the oracle circuit whose inputs are  $T$ -sets  $\{y_1, \dots, y_T\}$  (where  $y_1 < y_2 < \dots < y_T$ , lexicographically) and which runs as follows: On input  $(y_1, \dots, y_T)$ ,  $C'^R$  simulates, in order,  $C^R(y_1), \dots, C^R(y_T)$  with the only modification that if some simulated  $C^R(y_j)$  queries a string that has already been queried during the earlier simulations,  $C'$  will instead make a new query (on say the smallest lexicographically string that has not yet been queried). This is possible because  $ST \leq N$ . Thus the queries of  $C'$  are distinct. Let  $\{y_1, \dots, y_T\} \subseteq \{0, 1\}^n$  be a fixed  $T$ -set, with  $y_1 < \dots < y_T$ , and let  $q(1), q(2), \dots, q(S \cdot T)$  be the list in chronological order of the queries made by  $C'$  on input  $y_1, \dots, y_T$  in this order (we can assume without loss of generality that  $C$  makes exactly  $S$  queries on each input). Note that  $q(1), \dots, q(S \cdot T)$  are random variables that depend on  $R$  and on the random coins used during the simulations of  $C^R(y_1), \dots, C^R(y_T)$ . The probability that  $C^R$  inverts  $y_1, \dots, y_T$  is at most the probability that there exist  $i_1 \in [1, S \cdot T], i_2 \in [1, S \cdot T], \dots, i_T \in [1, S \cdot T]$ , such that  $q(i_1), \dots, q(i_T)$  are mapped by the function  $R$  onto, respectively,  $y_1, \dots, y_T$ . We fix  $T$  distinct integers  $i_1 \in [1, S \cdot T], i_2 \in [1, S \cdot T], \dots, i_T \in [1, S \cdot T]$  and we denote by  $A_{i_1, \dots, i_T}$  the event that  $q(i_1)$  maps into  $y_1, q(i_2)$  maps into  $y_2, \dots, q(i_T)$  maps into  $y_T$ . Since the queries  $q(i_1), \dots, q(i_T)$  are distinct and the random variables  $R(x)$ , for different  $x$ , are independent, it holds that,

$$\text{Prob}_R(A_{i_1, \dots, i_T} \text{ holds true} \mid \bar{E}) \leq \left(\frac{P}{N}\right)^T.$$

We note that the proof of the above claim (deferred for the final version) is more delicate than one first suspects (for instance, if the random variables  $R(x)$  would be only  $k$ -wise independent, with  $T < k \leq S \cdot T$ , then the claim would not hold).

There are  $(S \cdot T)_T$  tuples  $(i_1, \dots, i_T)$  as above, and, thus, the probability conditioned by  $\bar{E}$  that the fixed  $T$ -set  $\{y_1, \dots, y_T\}$  is inverted is at most  $(S \cdot T)_T \cdot \left(\frac{P}{N}\right)^T$ . Therefore, conditioned by  $\bar{E}$ , the expected number of  $T$ -sets that are inverted is at most  $\binom{N}{T} \cdot (S \cdot T)_T \cdot \left(\frac{P}{N}\right)^T$ . Let

$U = 2eP \cdot S \cdot T (= 2e\alpha \cdot N \cdot S \cdot T)$ . Then,

$$\begin{aligned}
\text{Prob}(\|\{x \mid C^R(R(x)) \in R^{-1}(R(x))\}\| \geq P \cdot U \mid \bar{E}) & \\
&\leq \text{Prob}(C^R \text{ inverts some } U\text{-set} \mid \bar{E}) \\
&\leq \text{Prob}(C^R \text{ inverts all } T\text{-subsets of some } U\text{-set} \mid \bar{E}) \\
&\leq \text{Prob}(\text{there are } \binom{U}{T} T\text{-sets that are inverted} \mid \bar{E}) \\
&\leq \frac{\binom{N}{T} \cdot (S \cdot T)^T \cdot \left(\frac{P}{N}\right)^T}{\binom{U}{T}} \quad (\text{by Markov's Inequality}) \\
&\leq \frac{(eN/T)^T \cdot (ST)^T \cdot (P/N)^T}{\left(\frac{U}{T}\right)^T} = \left(\frac{eP \cdot S \cdot T}{U}\right)^T = 2^{-T}.
\end{aligned}$$

(c) Let  $p$  be the probability that a random  $R$  is “good” for  $C$ . Then  $p \cdot \frac{1}{2}$  is  $\leq$  the fraction of the pairs  $(R, \rho)$  with the property that  $\|\{x \in \{0, 1\}^n \mid C^R(R(x), \rho) \in R^{-1}(R(x))\}\| \geq 2e \cdot \alpha N \cdot S \cdot T$ . The latter fraction, by point (b), is  $\leq 2^{-T} + 2^{-\Omega(n \cdot \sqrt{\alpha N})}$ . The conclusion follows.  $\blacksquare$

## 5 The extractor

There are three steps in the construction of a pseudo-random generator from a one-way length-preserving function  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . We follow these steps and analyze them keeping in mind that our goal is an exposure-resilient extractor, rather than a pseudo-random generator. We start with  $R(x)$ , for random  $x \in \{0, 1\}^n$ , and in STEP 1 we add  $\beta n$  *hidden bits*, for some parameter  $\beta$ . The hidden bits are given by a function  $a(x, r)$ , where  $r$  is an extra random input (which together with  $x$  is part of the seed). The key property is that the distributions  $(R(x), a(x, r), r)$  and  $(R(x), U_{\beta n}, r)$ , for random  $x$  and  $r$ , are statistically close. The function  $a(x, r)$  is obtained from an error-correcting code with appropriate list-decoding properties. Concretely, we are using a particular Reed-Muller code from [GRS98], in which we associate to  $x \in \{0, 1\}^n$  a certain multi-variable polynomial  $p_x$  over the field  $\text{GF}[2^{\beta n}]$ , and which has the property that if some function agrees with  $p_x$  on a fraction of  $1/2^{\beta n} + \epsilon'$  of the inputs, then, given black-box access to that function, one can produce in time  $\text{poly}(1/\beta, 1/\epsilon', \beta n)$  a list of elements that with probability at least  $3/4$  contains  $x$ . Thus, after STEP 1, we have  $(R(x), a(x, r), r)$  which “looks” similar to  $(R(x), U_{\beta n}, r)$ .

In STEP 2, we hash  $R(x)$  to  $H(R(x))$ , where  $H$  is a hash function randomly chosen from a universal family of hash functions mapping  $\{0, 1\}^n$  to  $\{0, 1\}^{n-(\beta n-1)}$ . By hashing we lose  $(\beta n - 1)$  bits, but, on the other hand, by the left-over hash lemma  $(H(R(x)), a(x, r), H, r)$  “looks” similar to  $(U_{n-(\beta n-1)}, U_{\beta n}, H, r)$ , which is the uniform distribution. Thus, after STEP 2, we have  $(H(R(x)), a(x, r), H, r)$  that looks similar to the uniform distribution and is one bit longer than  $(x, H, r)$  (which is the seed of the extractor).

In STEP 3, using the standard method of hybrid distributions, we increase the stretch.

**STEP 1.** - Adding hidden bits.

For a parameter  $\beta \in (0, 1)$ , we take  $q = 2^{\beta n}$  and consider the field  $\text{GF}[q]$ . We identify  $x \in \{0, 1\}^n$  with a polynomial  $p_x$  over the field  $\text{GF}[q]$  with  $b = 1/\beta$  variables  $y_1, \dots, y_b$  (to keep notation simple, we ignore truncation issues). This is done by breaking  $x$  into  $b$  blocks  $x_1, \dots, x_b$  of  $\beta n$  bits each, viewing each  $x_i$  as an element of  $\text{GF}[q]$ , and taking  $p_x(y_1, \dots, y_b) = x_1 y_1 + \dots + x_b y_b$ . For  $r \in [N]$ , we define  $\alpha_r$  to be the  $r$ -th element of  $(\text{GF}[q])^b$ . If we use binary notation then each  $r$  is  $n$  bits long. We define  $a : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \text{GF}[q]$  by  $a(x, r) = p_x(\alpha_r)$ . Note that  $a(x, r)$  is  $\beta n$  bits long.

**Lemma 5.1** Let  $C$  be a circuit with query complexity  $S$ . Let  $\epsilon > 0$ . Then there exists  $K = 2^{2\beta n}$  circuits  $A_1, \dots, A_K$  such that if for some  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$  it holds that

$$\text{Prob}_{x,r}[C^R(R(x), a(x,r), r) = 1] - \text{Prob}_{x,r,U_{\beta n}}[C^R(R(x), U_{\beta n}, r) = 1] > \epsilon \quad (2)$$

then there exist  $i \in [K]$  such that for more than a fraction of  $1/2$  of the strings  $\rho$  it holds that

$$\|\{x \mid A_i^R(R(x), \rho) \in R^{-1}(R(x))\}\| \geq \epsilon' \cdot N,$$

where  $\epsilon' = \frac{1}{2^{3q+1}}\epsilon$ . The circuits  $A_i$  have query complexity  $S \cdot \text{poly}(n, b, 1/\epsilon', q)$ .

**Proof** Let us fix a permutation  $R$ . Suppose the relation in Equation (2) holds. Then, adapting Yao's proof relating distinguishers to predictors, we show the following.

**Claim 5.2** There is a probabilistic circuit  $D$  with query complexity  $S$  such that

$$\text{Prob}[D^R(R(x) \odot r) = a(x,r)] > \frac{1}{2^{\beta n}} + \epsilon \cdot \frac{1}{2^{\beta n}},$$

where the probability is taken over random  $x, r$ , and the random coins of  $D$ .

**Proof** The circuit  $D^R$  on input  $R(x) \odot r$  works as follows:

1. Pick random  $u \in \{0, 1\}^{\beta n}$  and random  $u' \in \{0, 1\}^{\beta n}$ .
2. Simulate  $C^R(R(x), u, r)$ 
  - (a) If the simulation accepts (i.e.,  $C^R(R(x), u, r) = 1$ ), then return  $u$ .
  - (b) If the simulation does not accept, return  $u'$ .

Let  $E$  be the event (over random  $x, r, u$ , and  $u'$ ),

$$D^R(R(x) \odot r) = a(x,r).$$

Then,

$$\begin{aligned} \text{Prob}(E) &= \text{Prob}_{x,r,u}[C^R(R(x), u, r) = 1 \text{ and } u = a(x,r)] \\ &\quad + \text{Prob}_{x,r,u,u'}[C^R(R(x), u, r) = 0 \text{ and } u' = a(x,r)] \\ &= \text{Prob}_{x,r,u}[u = a(x,r)] \cdot \text{Prob}[C^R(R(x), u, r) = 1 \mid u = a(x,r)] \\ &\quad + \text{Prob}_{x,r,u,u'}[u' = a(x,r)] \cdot \text{Prob}[C^R(R(x), u, r) = 0 \mid u' = a(x,r)] \\ &= \text{Prob}_{x,r,u}[u = a(x,r)] \cdot \text{Prob}[C^R(R(x), a(x,r), r) = 1] \\ &\quad + \text{Prob}_{x,r,u,u'}[u' = a(x,r)] \cdot \text{Prob}[C^R(R(x), u, r) = 0]. \end{aligned}$$

Let  $p = \text{Prob}_{x,r}[C^R(R(x), a(x,r), r) = 1]$  and  $q = \text{Prob}_{x,r,u}[C^R(R(x), u, r) = 1]$ . Then,

$$\begin{aligned} \text{Prob}[E] &= \frac{1}{2^{\beta n}}(p + 1 - q) \geq \frac{1}{2^{\beta n}}(1 + \epsilon) \\ &= \frac{1}{2^{\beta n}} + \epsilon \cdot \frac{1}{2^{\beta n}}. \end{aligned}$$

The above Claim implies, using a standard Markov argument, that for at least a fraction of  $\epsilon' = \frac{1}{2^{\beta n+1}} \cdot \epsilon$  of  $x \in \{0, 1\}^n$ ,

$$\text{Prob}_{r,u,u'}[D^R(R(x), r) = a(x,r)] > \frac{1}{2^{\beta n}} + \epsilon' = \frac{1}{q} + \epsilon'.$$

We call an  $x$  verifying the above relation “good”. Note that the set of “good” strings depends on  $R$ . Thus, for a “good”  $x$ , we can fix in a convenient way the random strings  $u$  and  $u'$  such that  $D^R(R(x), r)$ , with the fixed values for  $u$  and  $u'$ , agrees with  $a(x, r) = p_x(r)$  on at least a  $(1/2^q) + \epsilon'$  fraction of  $r$ 's. Fixing  $u$  and  $u'$  in all possible ways, we get from  $D$  a number of  $K = 2^{2q}$  circuits  $D_1, \dots, D_K$  and, by the above discussion, for any “good”  $x$ , there is some circuit  $D_i$ ,  $i \in [K]$ , such that  $D_i^R(R(x), r)$  agrees with  $p_x(r)$  on at least a  $(1/2^q) + \epsilon'$  fraction of  $r$ 's. By the result from [GRS98] there is a probabilistic procedure that on input  $R(x)$  makes at most  $\text{poly}(n, b, 1/\epsilon', q)$  invocations of  $D_i(R(x), r)$  and produces a list with at most  $\text{poly}(n, b, 1/\epsilon', q)$  elements, and if  $D_i^R(R(x), r)$  agrees with  $p_x(r)$  as above, then, with probability greater than  $1 - 2^{-n}$ , the list contains  $x$ . By querying all the members of the list we obtain an element in the preimage set of  $R(x)$  (again in case  $D_i^R(R(x), r)$  agrees with  $p_x(r)$  as above). Note that the number of queries to  $R$  (needed during the invocations of  $D_i(R(x), r)$  and for querying the list members) is  $\text{poly}(n, b, 1/\epsilon', q) \cdot S$  (because  $D_i$  makes at most  $S$  queries). We denote this procedure  $A_i$ ,  $i \in [K]$ , and we say that  $A_i$  is an “inverter” for  $x$  if  $A_i^R(R(x), \rho) \in R^{-1}(R(x))$ , with probability of  $\rho$  at least  $1 - 2^{-n}$ . There are  $\frac{1}{2^{q+1}} \cdot \epsilon \cdot N$  “good”  $x$  and each one has an “inverter” in the set of  $K = 2^{2q}$  procedures  $A_i$ ,  $i \in [K]$ , defined above. Thus there must be one procedure  $A_i$  that is an “inverter” for at least a fraction of  $\frac{1}{2^{2q}} \cdot \frac{1}{2^{q+1}} \cdot \epsilon$  of  $x$  in  $\{0, 1\}^n$ . Let us fix this probabilistic procedure  $A_i$  and let  $J$  be the set of  $x \in \{0, 1\}^n$  for which  $A_i$  is an inverter. Thus  $\|J\| \geq \frac{1}{2^{3q+1}} \cdot \epsilon \cdot N$ . For each  $x \in J$ , the fraction of random strings  $\rho$  such that  $A_i^R(R(x), \rho) \notin R^{-1}(R(x))$  is  $\leq 2^{-n} = \frac{1}{N}$ . Let  $J'$  be a subset of  $J$  of size equal to  $\frac{1}{2^{3q+1}} \cdot \epsilon \cdot N$ . Thus, by the union bound, the fraction of strings  $\rho$  such that there exists some  $x \in J$  with  $A_i^R(R(x), \rho) \notin R^{-1}(R(x))$  is  $\leq \frac{1}{N} \cdot \frac{1}{2^{3q+1}} \cdot \epsilon \cdot N < \frac{1}{2}$ . ■

**STEP 2.** - Hashing.

We use a universal family of hash functions  $\mathcal{H}$ , where each  $H \in \mathcal{H}$  is of the type  $H : \{0, 1\}^n \rightarrow \{0, 1\}^{n-(\beta n-1)}$ , for some parameter  $\beta \in (0, 1)$ . For concreteness, we are using Toeplitz matrices. Each hash function is described by an  $(n - (\beta n - 1)) \times n$  Toeplitz matrix which can be described with a binary string that is  $(n - (\beta n - 1)) + n = 2n - (\beta n - 1)$  bits long. Abusing notation we denote by  $H$  both the hash function and its description as specified above. By the left-over hash lemma, if  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$  is a function that is  $P$ -to-1, for some positive integer  $P$ , then  $(H(R(U_n)), H)$  is  $\epsilon_1$ -statistically close to  $(U_{n-(\beta n-1)}, U_{2n-(\beta n-1)})$ , where  $\epsilon_1 = \sqrt{P/2^{\beta n-1}}$ .

**Lemma 5.3** *Let  $C$  be an oracle circuit with query complexity  $S$ . Then there are  $K = 2^{2n-(\beta n-1)}$  oracle circuits  $A_i$ ,  $i \in [K]$ , each with query complexity  $S$  such that if for some  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$  that is at most  $P$ -to-1 it holds that*

$$\text{Prob}_{x,r,H}[C^R(H(R(x)) \odot a(x,r) \odot H \odot r) = 1] - \text{Prob}[C^R(U_{n-(\beta n-1)} \odot U_{\beta n} \odot H \odot r) = 1] > \epsilon, \quad (3)$$

then for some circuit  $A_i$ ,  $i \in [K]$ ,

$$\text{Prob}_{x,r}[A_i^R(R(x) \odot a(x,r) \odot r) = 1] - \text{Prob}_{x,U_{\beta n},r}[A_i^R(R(x) \odot U_{\beta n} \odot r) = 1] > \epsilon - \epsilon_1. \quad (4)$$

**Proof** The left hand side in Equation (3) is equal to

$$\begin{aligned} & \text{Prob}_{x,r,H}[C^R(H(R(x)) \odot a(x,r) \odot H \odot r) = 1] - \text{Prob}[C^R(H(R(x)) \odot U_{\beta n} \odot H \odot r) = 1] \\ & + \text{Prob}[C^R(H(R(x)) \odot U_{\beta n} \odot H \odot r) = 1] - \text{Prob}[C^R(U_{n-(\beta n-1)} \odot U_{\beta n} \odot H \odot r) = 1]. \end{aligned}$$

The expression in the second line is bounded by  $\epsilon_1$ . Therefore, the expression in the first line is  $> \epsilon - \epsilon_1$ . We can build in the obvious way a probabilistic circuit  $B$  that chooses  $H$  randomly such that

$$\text{Prob}_{H,x,r}[B^R(R(x) \odot a(x,r) \odot r) = 1] - \text{Prob}_{H,x,U_{\beta n},r}[B^R(R(x) \odot U_{\beta n} \odot r) = 1] > \epsilon - \epsilon_1.$$

Fixing  $H$  in all possible ways we derive from  $B$  a number of  $K = 2^{2n-(\beta n-1)}$  deterministic circuits  $A_i$ ,  $i \in [K]$ , and one of them satisfies Equation (4). ■

**STEP 3.** - Stretching

We define the actual extractor EXT by the following algorithm.

Parameters:  $n \in \mathbf{N}$ ,  $m \in \mathbf{N}$ ,  $\beta \in (0, 1)$ .

Input: the weakly random string  $R \in \{0, 1\}^{n2^n}$ ,  $seed = x \odot r \odot H$ , where  $x \in \{0, 1\}^n$ ,  $r \in \{0, 1\}^n$ ,  $H \in \{0, 1\}^{2n-(\beta n-1)}$ .  $R$  is viewed as a function mapping  $\{0, 1\}^n$  to  $\{0, 1\}^n$  and  $H$  is viewed as a hash function given by a Toeplitz matrix as discussed above.

$b_0 =$  empty string;  $t_0 = x$ .

for  $i = 1$  to  $m$

$b_i =$  the first bit of  $H(R(t_{i-1})) \odot a(t_{i-1}, r)$ .

$t_i =$  the last  $n$  bits of  $H(R(t_{i-1})) \odot a(t_{i-1}, r)$ .

Output:  $b_1 \odot b_2 \odot \dots \odot b_m$  (i.e.,  $EXT(R, seed) = b_1 \odot b_2 \odot \dots \odot b_m$ ).

**Lemma 5.4** *For any oracle circuit  $C$  with query complexity  $S$ , there are  $K = 2^m$  oracle circuits  $A_i$ ,  $i \in K$ , each circuit  $A_i$  having query complexity  $S + m$ , such that if for some  $R$  and for some  $\epsilon > 0$ ,*

$$\text{Prob}_{x,r,H}[C^R(\text{EXT}(R, (x \odot r \odot H))) = 1] - \text{Prob}_{U_m}[C^R(U_m) = 1] > \epsilon, \quad (5)$$

then for some circuit  $A_i$ ,  $i \in [K]$ ,

$$\text{Prob}_{x,r,H}[A_i^R(H(R(x)) \odot a(x, r) \odot H \odot r) = 1] - \text{Prob}[A_i^R(U_{n-(\beta n-1)} \odot U_{\beta n} \odot H \odot r) = 1] > \frac{\epsilon}{m}.$$

**Proof** We fix a function  $R$  for which Equation (5) holds and we let  $B_i$ , for  $i \in [m]$ , be the distribution of the bit  $b_i$  in the algorithm  $\text{EXT}(R, (x \odot r \odot H))$ , when  $x$ ,  $r$ , and  $H$  are chosen uniformly at random. For  $k \in \{0, \dots, m\}$ , we define the distributions

$$D_k = U^1 \odot \dots \odot U^{m-k} \odot B_1 \odot B_2 \odot \dots \odot B_k,$$

where  $U^1, \dots, U^{m-k}$  are independent random bits. Note that  $D_0 = U_m$  and  $D_m = \text{EXT}(R, (x \odot r \odot H))$ . Since  $\text{Prob}[C^R(D_m) = 1] - \text{Prob}[C^R(D_0) = 1] > \epsilon$ , it follows that for some  $k \in \{0, \dots, m-1\}$

$$\text{Prob}[C^R(D_{k+1}) = 1] - \text{Prob}[C^R(D_k) = 1] > \epsilon/m. \quad (6)$$

We define the following procedure  $A$ .

Input:  $y \in \{0, 1\}^{n+1}$ ,  $H, r$ .

$y_1 =$  the first bit of  $y$ ;  $y_{\text{last}} =$  the last  $n$  bits of  $y$ .

$b_0 = y_1$ ;  $t_0 = y_{\text{last}}$ .

for  $j = 1$  to  $k$

$b_j =$  the first bit of  $H(R(t_{j-1})) \odot a(t_{j-1}, r)$ .

$t_j =$  the last  $n$  bits of  $H(R(t_{j-1})) \odot a(t_{j-1}, r)$ .

Randomly and independently choose  $m - k - 1$  random bits  $U^1, \dots, U^{m-k-1}$ .

Simulate  $C^R$  on  $U^1 \odot \dots \odot U^{m-k-1} \odot b_0 \odot b_1 \odot \dots \odot b_k$ . (\*)

Note that,

- (a) when  $y = U_{n+1}$ , the simulation of  $C^R$  in (\*) is on input  $D_k$ ,
- (b) when  $y = H(R(x)) \odot a(x, r)$ , the simulation is on  $D_{k+1}$ .

Thus, as a consequence of Equation (6), the procedure  $A$  distinguishes  $(H(R(x)) \odot a(x, r))$  from  $U_{n+1}$  with bias at least  $\epsilon/m$ . The procedure  $A$  makes  $S + k \leq S + m$  queries, and  $A$  uses  $m - k - 1 < m$  random bits. The circuits  $A_i$  are obtained from the procedure  $A$  by fixing the at most  $m$  random bits in all possible ways. ■

**Theorem 5.5** *Let  $\delta \in (0, 1)$ . There exists a polynomial-time computable  $(k, \epsilon)$ - exposure resilient extractor  $\text{EXT} : \{0, 1\}^{\tilde{n}} \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ , resistant to  $Q$  queries with the following parameters*

- (a)  $\tilde{n}$  is of the form  $n \cdot 2^n$ , for  $n \in \mathbf{N}$ ,
- (b)  $Q = \tilde{n}^\delta$ ,
- (c)  $k = \tilde{n} - \tilde{n}^{\Omega(1)}$ ,
- (d)  $\epsilon = \tilde{n}^{-\Omega(1)}$
- (e)  $d = O(\log \tilde{n})$ ,
- (f)  $m = \tilde{n}^{\Omega(1)}$ .

**Proof** We compose in order STEPS 1, 2, and 3, and the extractor is the procedure described in STEP 3.

The parameters required by the construction are taken as follows. We start with the given  $\delta \in (0, 1)$ . In view of Lemma 2.3, which doubles the bias, we need to work with  $\epsilon/2$ . For notation consistency with the previous lemmas, we let henceforth  $\epsilon$  abusively denote this value. The constant  $c$  will be specified later (it is derived essentially from the degree of the polynomial bounding the running time of the list-decoding procedure given in [GRS98]). Take  $s \in (\delta, 1)$ ,  $t \in (0, (1-s))$ ,  $\beta < \min(\frac{1-(s+t)}{(17/5)}, \frac{s-\delta}{c}, 5\delta)$   $\alpha = \min(\frac{1}{8e} \cdot 2^{-((17/5)\beta+s+t)n}, \frac{1}{64} 2^{-(1-(2/5)\beta)n})$ . Let  $m = 2^{(\beta/5)n}$ ,  $\epsilon = 2^{-(\beta/5)n}$ ,  $S = 2^{sn}$ ,  $T = 2^{tn}$ . Fix oracle circuit  $C$  with query complexity  $Q = \tilde{n}^\delta$ .

We estimate the number of functions  $R : \{0, 1\}^n \rightarrow \{0, 1\}^n$  that do not hit  $C^R$   $\epsilon$ -correctly. Let us consider such a function  $R$ . Let  $P = \sqrt{\alpha N}$ . The function  $R$  is either  $P - to - 1$  or it is not. Let us assume that it is  $P - to - 1$ .

By Lemma 5.4, there are  $K_1 = 2^m$  oracle circuits, denoted  $C_1, \dots, C_{K_1}$ , each one with query complexity  $Q + m$ , and for some  $i \in \{1, \dots, K_1\}$ , it holds that

$$\text{Prob}_{x,r,H}[C_i^R(H(R(x)) \odot a(r, x) \odot H \odot r) = 1] - \text{Prob}_{U_{n+1},r,H}[C_i^R(U_{n+1} \odot H \odot r) = 1] > \epsilon/m.$$

By Lemma 5.3, there are  $K_2 = 2^{2n-(\beta n-1)}$  oracle circuits, denoted  $C_{(i,1)}, \dots, C_{(i,K_2)}$ , each one with query complexity  $Q + m$  and for some  $j \in \{1, \dots, K_2\}$ , it holds that

$$\text{Prob}[C_{(i,j)}^R(R(x) \odot a(x, r) \odot r) = 1] - \text{Prob}[C_{(i,j)}^R(R(x) \odot U_{\beta n} \odot r) = 1] > \epsilon/m - \sqrt{P/2^{\beta n-1}} \geq \frac{\epsilon}{2m}.$$

Let  $\epsilon' = \frac{1}{2^{3\beta n+1}} \cdot \frac{\epsilon}{2m}$ . By Lemma 5.1, there are  $K_3 = 2^{2\beta n}$  oracle circuits, denoted  $C_{(i,j,1)}, \dots, C_{(i,j,K_3)}$ , each one with query complexity  $(Q + m)\text{poly}(n, n/q, 1/\epsilon', q)$  and one of them, denoted  $C_{(i,j,\ell)}$  has the property that for more than a fraction  $1/2$  of the strings  $\rho$

$$\|\{x \in \{0, 1\}^n \mid C_{(i,j,\ell)}^R(R(x), \rho) \in R^{-1}(R(x))\}\| \geq \epsilon' \cdot N.$$

In the expression  $\text{poly}(n, n/q, 1/\epsilon', q)$ , the dominant term is  $1/\epsilon'$  and it can be seen that there is a constant  $c$ , which we mentioned above, such that  $\text{poly}(n, n/q, 1/\epsilon', q) \leq 2^{c\beta n}$ . By the way the parameters were chosen, it holds that the query complexity of  $C_{(i,j,\ell)}$  is less than  $S$  and  $\epsilon' \cdot N \geq (2e \cdot \alpha N \cdot S \cdot T)$ . Recall from Lemma 4.1 that we say that  $R$  is “good” for a probabilistic oracle circuit  $C$  if for more than a fraction  $1/2$  of strings  $\rho$ ,  $\|\{x \in \{0, 1\}^n \mid C^R(R(x), \rho) \in R^{-1}(R(x))\}\| \geq 2e \cdot \alpha N \cdot S \cdot T$ . Therefore if  $R$  does not hit  $C_4$   $\epsilon$ -correctly then either (a)  $R$  is not  $P - to - 1$ , or (b)  $R$  is “good” for the circuit  $C_{(i,j,\ell)}$ , one of the  $K_1 \cdot K_2 \cdot K_3$  circuits that result from the composition of the three steps.

By Lemma 4.1, the number of such  $R$ 's is bounded by  $2^{nN} \cdot 2^{-\Omega(n \cdot \sqrt{\alpha N})} + K_1 \cdot K_2 \cdot K_3 \cdot 2 \cdot 2^{nN} (2^{-T} + 2^{-\Omega(n \cdot \sqrt{\alpha N})}) \leq 2^{nN - N^{\Omega(1)}}$ . Thus by Lemma 2.3, the min-entropy of the extractor is  $nN - N^{\Omega(1)} + \log(\frac{1}{\epsilon})$ . ■

## References

- [BST03] B. Barak, R. Shaltiel, and E. Tromer. True random number generators secure in a changing environment. In *Workshop in Cryptography Hardware and Embedded Systems (CHES)*, number 2779 in LNCS, pages 166–180. Springer Verlag, 2003.
- [Dod00] Y. Dodis. *Exposure-resilient Cryptography*. PhD thesis, MIT, August 2000.
- [DSS01] Y. Dodis, A. Sahai, and A. Smith. On perfect and adaptive security in exposure-resilient cryptography. In *Advances in Cryptology - EUROCRYPT*, May 2001.
- [GRS98] O. Goldreich, R. Rubinfeld, and M. Sudan. Learning polynomials with queries - the highly noisy case. Technical Report TR98-060, Electronic Colloquium on Computational Complexity, 1998. <http://www.eccc.uni-trier.de/eccc>.
- [GT00] R. Gennaro and L. Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, 2000.
- [HILL99] J. Håstad, R. Impagliazzo, L. Levin, and M. Luby. Construction of a pseudo-random generator from any one-way function. *SIAM Journal on Computing*, 28(4), 1999.
- [Imp96] R. Impagliazzo. Very strong one-way functions and pseudo-random generators exist relative to a random oracle. (manuscript), January 1996.
- [KZ03] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 92–101. IEEE, 2003.
- [Lu04] C.J. Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, 17(1):27–42, January 2004.
- [Nis96] N. Nisan. Extracting randomness: how and why. A survey. In *Proceedings of the 11th Structure in Complexity Theory Conference*, pages 44–58, 1996.
- [Sha02] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin EATCS*, 77:67–95, June 2002.
- [Tre01] L. Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.
- [Vad] S. Vadhan. Randomness extractors and their cryptographic applications. Netherlands RISC Seminar and Intercity Number Theory Seminar, January 21, 2005. Available at <http://eecs.harvard.edu/salil/survey-talks.html>.
- [Vad04] S. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. of Cryptology*, 17(1):43–77, January 2004.
- [Wee05] H. Wee. On obfuscating point functions. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 523–532, May 2005.
- [Zim05] M. Zimand. Simple extractors via constructions of cryptographic pseudo-random generators. In *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science 3580, pages 115–127. Springer-Verlag, 2005.