# The Trevisan Extractor

Johan Wallén

Helsinki University of Technology
Laboratory for Theoretical Computer Science

johan@tcs.hut.fi

## Introduction

We will give a detailed description of the extractor from Luca Trevisan, Extractors and Pseudorandom Generators, *Journal of the ACM* 48(4):860–879, 2001. (Extended abstract in the 31st ACM Symposium on Theory of Computing, 1999.)

Our presentation is based on Miltersen's survey.

We will first recall the definition of extractors and present the main building blocks, namely source encodings and Nisan-Wigderson designs.

We will then present the extractor and prove its correctness.

Finally, we take a look at one concrete derandomisation result that can be obtained using Trevisan's extractor.

## Extractors

Recall that the **min-entropy** of a probability distribution on a finite domain is at least $t$ if every outcome has probability at most $2^{-t}$.

Two distributions $D_1$ and $D_2$ on a finite domain are $\epsilon$**-close** if their $L_1$-distance it at most $\epsilon$,

$$\|D_1 - D_2\|_1 = \sum_x \left| \Pr_{D_1}[x] - \Pr_{D_2}[x] \right| \leq \epsilon.$$

## Extractors (cont.)

An extractor is a device that extracts the randomness from weak random sources.

An **extractor** $E$ with **min-entropy threshold** $t$ and **error** $\epsilon$ is a function $E\colon U \times A \to V$, such that the distribution of $E(x, y)$ is $\epsilon$-close to the uniform distribution on $V$ when $x$ is selected from a distribution with min-entropy at least $t$ and $y$ is selected uniformly at random.

The extractor $E$ is **explicit**, if $|U| = 2^n$ and $|A| = 2^k$, and there is an algorithm that on input $(x, y)$ runs in time polynomial in $n$ and $k$ and outputs $E(x, y)$.

Recall that an extractor can be seen as a bipartite graph with vertices $(U, V)$ and edges labelled by $A$.

## The main theorem

The main theorem is the following (Theorem 9 in Miltersen's survey):

For any constants $k > c > 1$, there is an explicit extractor

$$E\colon \{0,1\}^{r^k} \times \{0,1\}^{O(\log r + \log \epsilon^{-1})} \to \{0,1\}^r$$

for any parameter $r$, with error $\epsilon$ and min-entropy threshold $r^c$.

Actually, we will only consider the case $c = 3$ and $\epsilon = 1/10$ (this is enough for most constructions).

We will give a more precise form of the result later.

The polynomial running time is unfortunately large.

## Source encodings

One way to prove the correctness of an extractor is to assume that the output distribution is not close to uniform and show that the min-entropy of the source is small.

To do this, we need a good way to show that a distribution has a small min-entropy.

By a classical result, the Shannon entropy of a source is a lower bound on the expected length of the codewords in every lossless source code.

If we instead use lossy source codes (combined with error correction), we can get a similar result for the min-entropy.

## Lossy source encodings

A **lossy** fixed length **source code** with **rate** $k$ is a pair of functions, the encoder $c\colon \{0,1\}^n \to \{0,1\}^k$ and the decoder $d\colon \{0,1\}^k \to \{0,1\}^n$.

Note that we do not require that $d(c(x)) = x$.

Given any distribution $D$ for $x$, we define the **distortion** of the source code relative to $D$ by

$$\Pr[d(c(x))_i \neq x_i],$$

where $x$ is selected according to $D$ and $i$ is uniformly distributed in $\{1, \dots, n\}$.

## Lossy source encodings (cont.)

We cannot obtain the result we want by constructing a lossy source code with low rate and low distortion.

There are distributions with high min-entropy that admit lossy source codes with low rate and low distortion (for example, the JPEG standard is based on this observation).

But if we first apply a good error correcting code to our distribution, and then consider lossy source codes for the encoded distribution, we do get the desired result.

## Error correcting codes

Let $\mathbf{F}$ be a finite field. Recall that a **linear error correcting code** with **relative minimum distance** $\delta$ is a linear mapping $e\colon F^n \to F^m$, $m > n$, such that the fraction of entries where $e(x)$ and $e(y)$ differs is at least $\delta$ for all $x \neq y$.

That is,

$$\frac{d_h(x,y)}{n} = \frac{|\{i \mid e(x)_i \neq e(y)_i\}|}{n} \geq \delta$$

for all $x \neq y$.

If $e(x)$ has the form $e(x) = (x, x')$, we say that the code is **systematic**.

If $|\mathbf{F}| = 2$, we call $e$ a **Boolean** code.

## Error correcting codes (cont.)

By concatenating a Reed-Solomon code with a Hadamard code with suitable parameters, we get the following result (Lemma 12 in Miltersen's survey).

Let $n$ be a power of $2$. There is a polynomial (in $n$) time computable, linear, systematic, Boolean code $e\colon \{0,1\}^n \to \{0,1\}^{n^{10}}$ with relative minimum distance at least $1/2 - 1/n^4$.

We omit the proof (it is not difficult, but we would have to take a detour to explain it).

## Main lemma for lossy source codes

The following lemma (Lemma 25 in Miltersen's survey) is crucial.

Let $n$ be a power of $2$, and let $e\colon \{0,1\}^n \to \{0,1\}^{n^{10}}$ be the error correcting code on the previous slide. Let $D$ be a distribution on $\{0,1\}^n$ with min-entropy at least $t$. Let $e(D)$ be the induced probability distribution on $\{0,1\}^{n^{10}}$.

Then every lossy fixed length source code for $\{0,1\}^{n^{10}}$ with distortion less than $1/2 - 2/n^2$ relative to $e(D)$ has rate at least $t - 6\log_2 n$.

(Note that the relative distortion $1/2$ is trivial.)

## Main lemma for lossy source codes: proof

Let $(c, d)$ be the source code, and let $k$ be its rate. Let $G$ be the event that a sample $y$ from $e(D)$ has Hamming distance from $d(c(y))$ smaller than $1/2 - 1/n^2$.

Then $\Pr[y \in G] \geq 1/n^2$, since otherwise the distortion would be more than $(1 - 1/n^2)(1/2 - 1/n^2) > 1/2 - 2/n^2$.

Since the rate of the source code is $k$, there are at most $2^k$ code words. Let $w$ be the most frequent value of $c(y)$ given that $y \in G$. Let $G'$ be the event that $y \in G$ and $c(y) = w$. Clearly, $\Pr[y \in G'] \geq 1/(2^k n^2)$.

## Main lemma for lossy source codes: proof (cont.)

On the other hand, all $y \in G$ are $e$-code words, and if $c(y) = w$, $y$ is in the Hamming ball with centre $d(w)$ and relative radius $1/2 - 1/n^2$ (the distortion of the source code).

By a result on list decoding (see Lemma 14 in Miltersen's survey), this Hamming ball can contain at most $n^4/2$ code words of $e$. Thus, there are at most $n^4/2$ outcomes in $G'$. Since the min-entropy of $e(D)$ is at least $t$, each outcome has probability at most $2^{-t}$. Hence, $\Pr[G'] \leq 2^{-t}n^4/2$.

It follows that $1/(2^k n^2) \leq \Pr[G'] \leq 2^{-t}n^4/2$ so that $k \geq t - 6\log_2 n$.

## Nisan-Wigderson designs

A Nisan-Wigderson design is a set systems with small pairwise intersections. More precisely, an $(m, n, s, \ell)$**-Nisan-Wigderson design** consists of $m$ subsets $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ of size $s$ such that $\left|S_i \cap S_j\right| \leq \ell$ for all $i \neq j$.

Explicit Nisan-Wigderson designs can easily be constructed using weakly universal hash functions, but this construction is unfortunately not good enough.

Fortunately, we only need a polynomial-time algorithm that constructs a suitable Nisan-Wigderson design.

## Constructing Nisan-Wigderson designs

The following lemma (Lemma 27 in Miltersen’s survey) is crucial.

For any integer constant $c \geq 1$, and every $m$ that is a power of $2$, there is a deterministic algorithm that on input $m$ outputs an $(m, 100c^2 \log m, c \log m, \log m)$-Nisan-Wigderson design, using time polynomial in $m$.

The polynomial is quite large: the straightforward bound is

$$c \log m \cdot m^2 \binom{100c^2 \log m}{c \log m}$$

operations. Note that this is polynomial in $m$, since $c$ is a constant.

## Constructing Nisan-Wigderson designs: proof

We use a simple “greedy” algorithm. Suppose that we have already picked the subsets $S_1, \ldots, S_i$ of the correct size and satisfying the constraint on the size of the pairwise intersections.

If we can show that there exists a set $S_{i+1}$ of the correct size that can be added to the set system without violating the constraint, we have our polynomial-time algorithm, since we can exhaustively search through all subsets of the correct size (this is where the large factor comes from).

Pick randomly with replacement $2c \log m$ members of $\{1, \ldots, 100c^2 \log m\}$. Let $S$ be the resulting multiset. Using the Chernoff bound, it can be shown that with high probability, $S$ contains at least $c \log m$ disjoint elements and the intersection of $S$ with any of the previous sets is less than $\log m$.

## The Trevisan extractor

The Trevisan extractor is a polynomial-time computable map $E\colon U \times A \to V$, where $U = \{0,1\}^{r^k}$, $A = \{0,1\}^{c \log r}$, $V = \{0,1\}^r$, $k$ is any integer constant greater that $5$, and $c = 10^4 k^2$.

We assume that all parameters are such that all numerical values in the construction are integers.

We will show that for sufficiently large $r$, the extractor $E$ (described on the next slide) has error at most $1/10$ and min-entropy threshold at least $r^3$.

## The Trevisan extractor (cont)

We describe $E$ as an algorithm.

Given the parameter $r$, the extractor uses the previous lemma to construct an $(r, c\log r, 10k\log r, \log r)$-Nissan-Wigderson design $S_1, \ldots, S_r$.

Let $e\colon \{0,1\}^{r^k} \to \{0,1\}^{r^{10k}}$ be the error correcting code mentioned earlier.

Let $x \in U$ be the first input of $E$. The extractor computes the value $e(x)$. We represent the sequence of bits $e(x)$ by its characteristic function

$$g\colon \{0,1\}^{10k\log r} \to \{0,1\}.$$

## The Trevisan extractor (still cont.)

Let $y \in A$ be the second input of $E$. Each set in the design in a subset of $\{1, \ldots, c \log r\}$. We view $y$ as a bit sequence of length $c \log r$.

For each set in the design, let $y_S$ be the subsequence of the bits indexed by $S$. For example, if $y = 10111001$ and $S = \{2, 4, 7\}$, $y_S = 010$.

The output of the extractor is $E(x, y) = g(y_{S_1})g(y_{S_2}) \cdots g(y_{S_r})$.

That is, the extractor uses the design to select subsequences of the uniformly distributed input. Then the characteristic function of the encoded version of the input from the weak random source is used to obtain one bit from each subsequence.

## Correctness proof

It remains to show that the extractor $E$ has error $1/10$ and min-entropy threshold $r^3$.

Let $D$ be a distribution on $U = \{0,1\}^{r^k}$ with min-entropy at least $r^3$. Pick $x$ according to $D$ and $y \in A$ uniformly at random.

Suppose that $E(x, y)$ is not $1/10$-close to the uniform distribution.

By a simple result (Lemma 21 in Miltersen's survey), there is is a predictor with advantage $\epsilon = 1/20r$ for one of the bits of $E(x, y)$, say the $i$th, given the previous bits. That is, the probability that the predictor outputs $E(x, y)_i$ given $E(x, y)_1, \ldots, E(x, y)_{i-1}$ is at least $1/2 + \epsilon$.

## Correctness proof (cont.)

We will use the predictor to construct a good lossy source code $(c, d)$ for $e(x)$ and thus reach a contradiction.

The $i$th bit of $E(x, y)$ is $g(y_{S_i})$. The input to the predictor is $g(y_{S_1}) \cdots g(y_{S_{i-1}})$.

For the moment, we will set the bits $y_{S_i^c}$ (that is, the bits not in $y_{S_i}$) to fixed values. Let $j < i$. Since $\left|S_j \cap S_i\right| \leq \log r$ and all the bits in $y_{S_j \setminus S_i}$ have been fixed, we can for fixed $x$ (and hence $g$) tabulate the values $g(y_{S_j})$ indexed by $S_j \cap S_i$ as a bitstring of length $2^{\log r} = r$.

We define $c(e(x))$ to be the concatenation of these $r$ tables and the binary representation of $i$. The rate of this source code (with additional padding) is $r^2 + \log r$.

## Correctness proof (still cont.)

Given a code word $z = c(e(x))$, we go through all the possible values of $y_{S_i}$, and use the tables in the code word to look up the corresponding values of $g(S_1), \ldots, g(S_{i-1})$. We give these values to the predictor to get a prediction of $g(S_i)$.

Since we try all possible values of $S_i$, we get predictions for all bits of $e(x)$ (the correct $g(y_{S_i})$ is the value of $e(x)$ indexed by $S_i$). We output the concatenation of the predicted values of $e(x)$ as $d(c(e(x)))$.

## Correctness proof (still cont.)

Since the predictor has advantage $\epsilon$, the probability that $d(c(e(x)))_j \neq e(x)_j$ is at most $1/2-\epsilon$, when $x$ is selected according to $D$ and $j$ is uniformly distribution.

That is, the distortion is at most $1/2 - \epsilon = 1/2 - 1/20r$.

By the lemma earlier, the rate of the source code must be at least $r^3 - 3\log 20r$. This is a contradiction, since the rate was already seen to be $r^2 + \log r$.

Thus, the output of the extractor must be $1/10$-close to uniform.

## Randomness efficient amplification

We have earlier seen results of the type “if we have a good extractor, we can decrease the error probability of an algorithm a lot without using many extra random bits”. We will use the Trevisan extractor to obtain an unconditional result of this type.

Suppose that we have a two-sided error algorithm that runs in time $T$ and uses $r$ random bits to achieve an error probability of $1/3$.

Let $E\colon \{0,1\}^{r^k} \times \{0,1\}^{c_k \log r} \to \{0,1\}^r$ be the Trevisan extractor with error at most $1/10 < 1/3$ and min-entropy threshold at least $r^3$. Let the running time of $E$ be $t_k(r)$. (Here, $k$ is a parameter.)

## Randomness efficient amplification (cont.)

There is an two-sided error algorithm running in time $(t_k(r)+T)^{r^{c_k}}$ that uses $r^k$ random bits to achieve an error probability of $2^{r^3-r^k}$. (This is a concrete version of Lemma 22 in Miltersen's survey.)

Let $x \in \{0,1\}^{r^k}$ be the random bits of the simulator, and let $y_1, \ldots, y_{r^{c_k}} \in \{0,1\}^r$ be all the possible outputs of the Trevisan extractor when the first input is $x$.

The simulator simulates the original algorithm on all the coin toss sequences $y_i$ and outputs the majority outcome.

## Randomness efficient amplification: correctness

By symmetry, we can assume that the correct output is “accept”. Then at least $2/3$ of the strings in $\{0,1\}^r$ corresponds to accepting computations of the original algorithm.

We claim that less than $2^{r^3}$ of the random inputs $x$ to the simulator make it reject.

Suppose not. Then the uniform distribution on the choices of $x$ that causes reject have min-entropy at least $r^3$. If we pick such an $x$, and the second input to the extractor uniformly at random, we obtain a value $y \in \{0,1\}^r$ that is $1/10$-close to uniform.

## Randomness efficient amplification: correctness (cont.)

Recall that the statistical distance between two distributions $D_1, D_2$ on a finite domain $A$ can be expressed as

$$\|D_1 - D_2\|_1 = \sum_{d \in A} \left| \Pr_{D_1}[d] - \Pr_{D_2}[d] \right| = \frac{1}{2} \max_{T \subseteq A} \left| \Pr_{D_1}[d \in T] - \Pr_{D_2}[d \in T] \right|.$$

Let $T \subseteq \{0,1\}^r$ consist of all accepting coin tosses of the original algorithm. If $y$ is uniformly distributed, $\Pr[y \in T] \geq 2/3$ (since the correct output is to accept).

If $y$ is selected using the extractor from the rejecting coin tosses for the simulator, $\Pr[y \in T] > 1/2$, since $y$ is $1/10$-close to uniform. This is a contradiction, since a minority of the values $y$ causes a reject.