Simple extractors via crypto pseudo-random generators

Marius Zimand

(Towson University)

Extractors vs. Pseudo-Rand. Generators

- Fundamental primitives in derandomization, cryptography, ...

- They "manufacture" randomness, but different sort of randomness

- Extractor = procedure that transforms imperfect randomness into (almost) perfect randomness (information-theoretic randomness).

- P.R. Gen. = procedure that transforms a short random seed into a long string that looks random to any poly-time adversary (computational randomness).

- (OLD?) Common Wisdom: information-theoretic randomness and computational randomness live in two very different worlds.

Trevisan's result ('99)

- There are two types of p.r.generators
  - p.r. gen. constructed from a hard function (Nisan-Wigderson) $\rightarrow$ derandomization
  - p.r. gen constructed from a one-way permutation/function (Blum-Micali-Yao) $\rightarrow$ crypto
- Trevisan'99: the N-W construction $\Rightarrow$ extractors
- " ... it might also be that our results are just an isolated exception to the "rule" that computational randomness results are not useful in information theoretic settings. For ex., we note that the p.r. gen constructions of B-M-Y does not yield an extractor using our technique."

- **Our result:** B-M-Y method $\Rightarrow$ extractors

- ... and they are very simple (locally computable!).

Quality of Randomness

Perfect Randomness = Unif. distribution

n-bit strings

- each string is equally likely to be produced
- for all x, $\Pr(X = x) = 2^{-n}$

Imperfect randomness = min-entropy $< n$

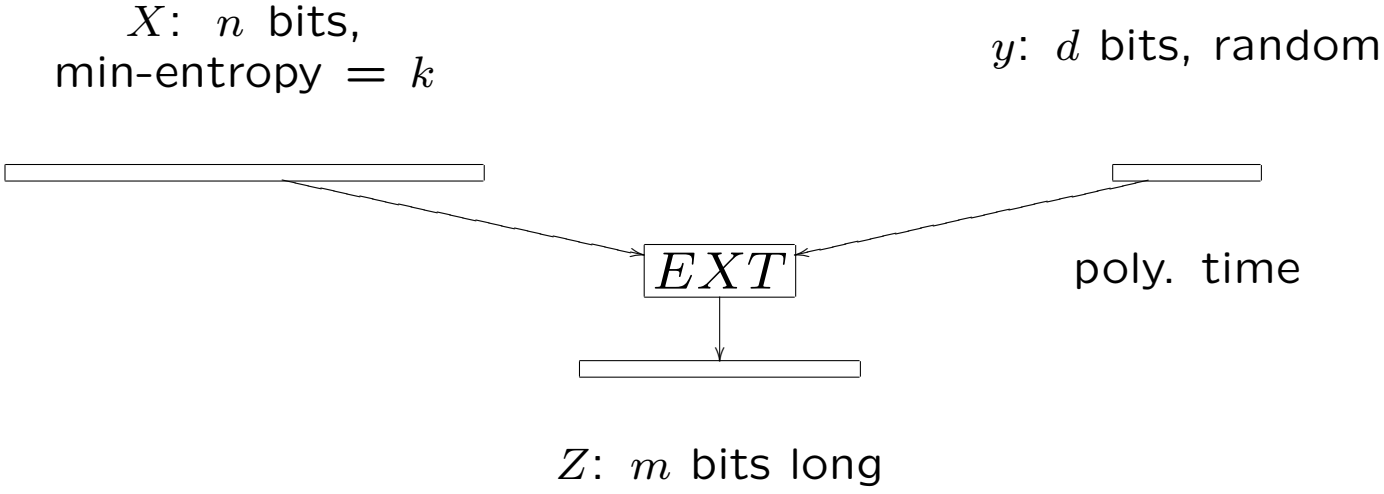
n-bit strings

- some strings are more likely than other strings
- for all x, $\Pr(X = x) \leq 2^{-k} \stackrel{def}{\Leftrightarrow} \min - \text{entropy}(X) = k$

Definition of extractor

Extractors = procedures that transform imperfect randomness into perfect randomness

We consider seeded extractors



- $\Delta_{\text{stat}}(Z, U_m) < \epsilon$
- For any $A \subseteq \{0,1\}^m$, $|\Pr(Z \in A) - \frac{|A|}{2^m}| < \epsilon$
- We say that EXT : $\{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ)-extractor.

Methods for constructing extractors:

- '89 – '99: Hashing (left-over hash lemma) + sampling

- '99: Trevisan

  Nisan-Wigderson method for p.r.-generators + error-correcting codes

  $\rightarrow$ extractors

- post '99: combinations and refinements of the above

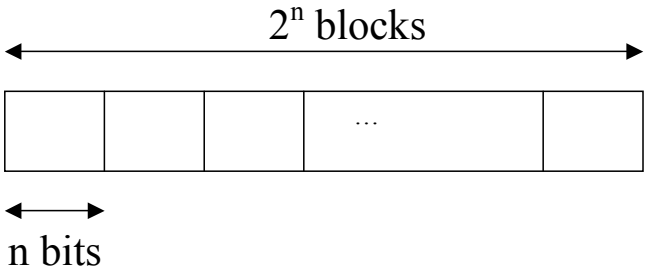
- Here:

  Blum-Micali-Yao methods for p.r.-generators

  $\rightarrow$ extractors

Our extractors are very simple

X:



seed $= ((y_1, y_2, \ldots, y_\ell), r)$, $|y_i| = n$, $|r| = \ell n$, $\ell = O(n)$

$b_0 = (X[y_1] \ldots X[y_\ell]) \cdot r$ (inner prod. mod 2)

$b_1 = (X[y_1 + 1] \ldots X[y_\ell + 1]) \cdot r$

$\ldots$

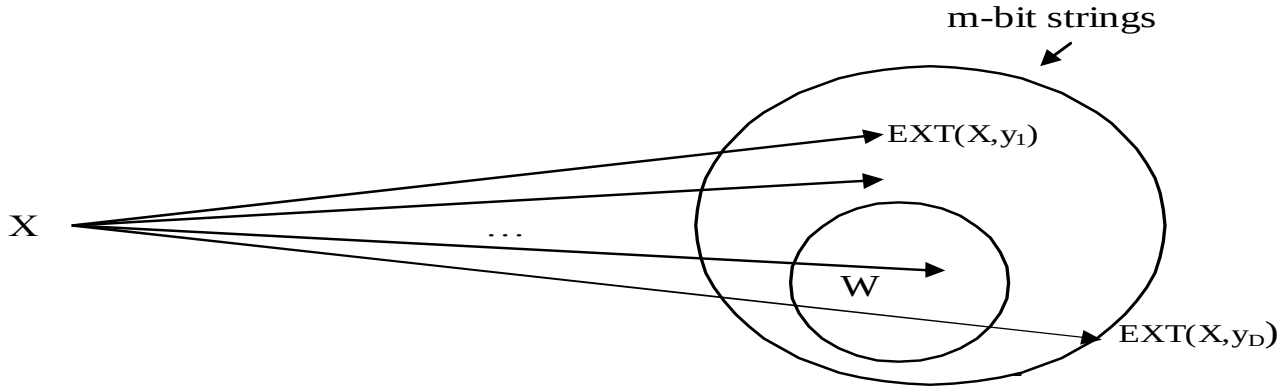
$b_{m-1} = (X[y_1 + m - 1] \ldots X[y_\ell + m - 1]) \cdot r$

output $= b_0 b_1 \ldots b_{m-1}$

Parameters (not optimal but non-trivial)

- Notation: $|X| = N = (n \cdot 2^n)$
- Locally computable extractor: each output bit is extracted in $O(\log^2 N)$ time.
- Works for min-entropy λN, any const. λ
- Seed length $O(\log^2 N)$
- Output length $\approx N^{\lambda/3}$
- Also seed length $O(\log N)$ but output length $2^{c\sqrt{\log N}}$

A useful lemma



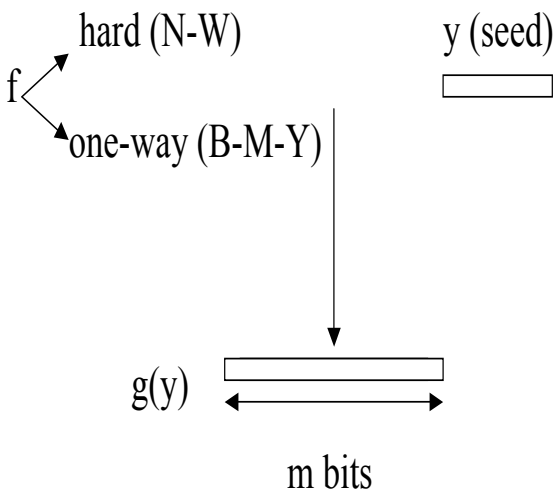
DEF. X hits W ϵ-correctly if

$$\left|\frac{|\{y \mid \mathsf{EXT}(X,y) \in W\}|}{D} - \frac{|W|}{2^m}\right| < \epsilon$$

Lemma. If $\forall W$, no. of X that do not hit W ϵ-correctly is $\leq 2^t$

$\Rightarrow$ EXT is a $(t + \log(1/\epsilon), 2\epsilon)$-extractor

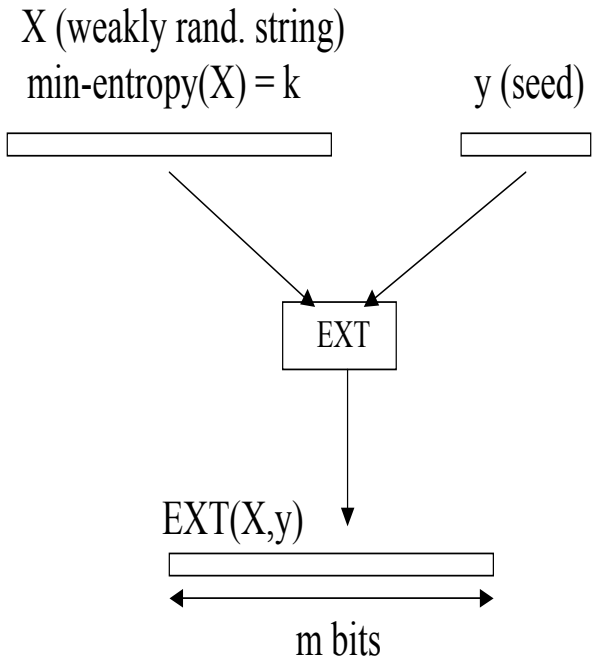
Pseudo-rand. generator



for all $A \subseteq \{0,1\}^m$, computable by poly-size circuits,

$$\left|\Pr(g(y) \in A) - \frac{|A|}{2^m}\right| < \epsilon$$

Extractor



for all A,

$$\left|\Pr(\mathsf{EXT}(X,y) \in A) - \frac{|A|}{2^m}\right| < \epsilon$$

Trevisan's method

$$f \quad \overset{\text{N}-\text{W}}{\longrightarrow} \quad g_f$$

hard-function p.r.gen.

N-W: If circuit D distinguishes $g_f(y)$ from unif., one can transform D into A, not much larger, and A computes f

Trevisan:

- Take a random f
- View the truth-table of f as the weakly random string
- $\mathsf{EXT}(f, y) = g_f(y)$

- f does not hit D ϵ-correctly $\Leftrightarrow$ D disting. f from unif.

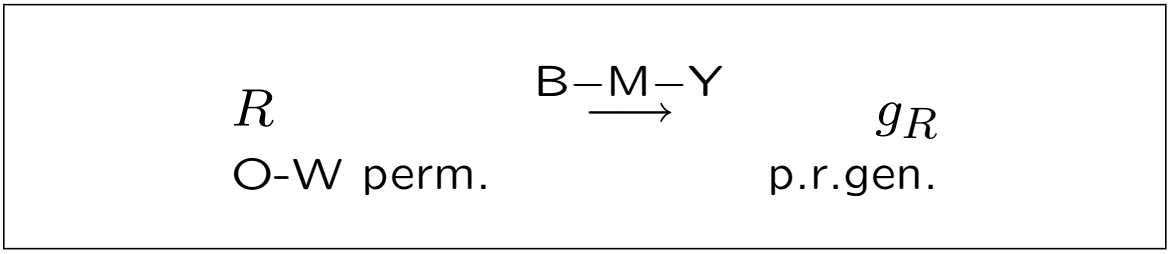
- f can be calculated by small A with D-gates

- f has a small description, so there are few f's like this

- $|\{f \mid f$ does not hit D ϵ-correctly$\}|$ is small

- So, by Lemma, EXT is an extractor

B-M-Y: p.r.gen. from one-way permutations



B-M-Y: If circuit D distinguishes $g_R(y)$ from unif., one can transform D into A, not much larger, and A inverts R on may inputs.

Try to mimic Trevisan:

- R does not hit D ϵ-correctly $\Leftrightarrow \exists A$, built from D, distinguisher.

- We need:

  Most permutations are one-way. (????)

- Solution: Allow circuits that calculate R to have oracle access to R, permutation on n-bit strings.

- All R's are easy to calculate.

- R does not hit D ϵ-correctly $\Rightarrow \exists A$ with query complexity poly$(m, 1/\epsilon)$ that inverts R on $\epsilon/$poly$(m, 1/\epsilon)$ fraction of $R(x)$'s.

- We need: for decently long m, the number of such R's is small.

- Known techniques (Impagliazzo'96, Gennaro-Trevisan'00, Wee'05) are not enough.

- Closer look at B-M-Y: D distinguisher

  $\Rightarrow A$ (built from D) is a **'nice'** inverter:

  determines x using only the values of $R(x), R^2(x), \ldots, R^m(x)$ (m is the output length).

- Intuitively, for a random perm R, x should be almost indep. of $R(x), R^2(x), \ldots, R^m(x)$.

- We restrict to cyclic permutations $1 \rightarrow R(1) \rightarrow R^2(1) \rightarrow \ldots \rightarrow R^{N-1}(1) \rightarrow 1$.

- Using A, we can reconstruct perm. R only from $R^{N-1}(1), R^{N-2}(1), \ldots, R^{N-m}(1)$

- How: $R^{N-1}(1), R^{N-2}(1), \ldots, R^{N-m}(1) \Rightarrow R^{N-m-1}(1)$ $R^{N-m-2}(1) \Rightarrow \ldots \Rightarrow R(1)$.

So, R does not hit D ϵ-correctly

$\Rightarrow$ D distinguisher

$\Rightarrow$ $\exists A$ 'nice' inverter

$\Rightarrow$ R has small description

$\Rightarrow$ few such R's

$\Rightarrow$ extractor.

several complications ...

- the weakly random string is not a cyclic permutation.
  $\rightarrow$ we convert it into one.

- A does not determine x from $R(x), R^2(x), \ldots,$ $R^m(x)$, but only finds a list containing x.
  $\rightarrow$ we include in the description the rank of x in the list.

- The above holds only for ϵ/m fraction of x.
  $\rightarrow$ amplify this to $(1-\delta)$ using the "weak-O-W to strong-O-W" transformation.

- A is a randomized circuit and we cannot hardwire the "best" bits (they may be different for different R's).
  $\rightarrow$ we get more inverting circuits, and the index of the right one must be included in the description of R.

Conclusions/Open Problems:

- The Blum-Micali-Yao technique works for extractors.

- There are tight lower bounds for extractors. Can they be used to obtain lower bounds for the various steps in B-M-Y method?

- Extractors obtained in this way are very simple and some are locally computable (each bit is extracted separately in polylog time).

- Parameters are not optimal at this time. Can they be improved?

- Maybe good for teaching extractors (after B-M-Y has been covered).