A PSEUDORANDOM GENERATOR CONSTRUCTION
BASED ON
RANDOMNESS EXTRACTORS AND COMBINATORIAL
DESIGNS

by

Ladan A. Mahabadi
B.Sc., University of British Columbia, Vancouver, Canada, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

© Ladan A. Mahabadi 2006
SIMON FRASER UNIVERSITY
Spring 2006

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.
APPROVAL

Name: Ladan A. Mahabadi

Degree: Master of Science

Title of thesis: A Pseudorandom Generator Construction based on Randomness Extractors and Combinatorial Designs

Examining Committee: Dr. Arthur Kirkpatrick
Chair

Dr. Arvind Gupta, Supervisor,
Professor of Computing Science
Simon Fraser University

Dr. Valentine Kabanets, Senior Supervisor,
Professor of Computing Science
Simon Fraser University

Dr. Gábor Tardos, Examiner,
Professor of Computing Science,
Simon Fraser University

Date Approved:  Apr.-13/06
DECLARATION OF
PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection, and, without changing the content, to translate the thesis/project or extended essays, if technically possible, to any medium or format for the purpose of preservation of the digital work.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright License.

The original Partial Copyright License attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

Simon Fraser University Library
Burnaby, BC, Canada
Abstract

Nisan and Wigderson in their seminal work introduced a new (conditional) pseudorandom generator construction which since then has been extensively used in complexity theory and has led to extensive further research. Impagliazzo and Wigderson (1997), and Sudan, Trevisan, and Vadhan (2001) have shown how this construction can be utilized to prove conditional derandomization results under weaker hardness assumptions. We study the construction of pseudorandom generators, and use an observation of Sudan et al. to recast the Impagliazzo-Wigderson construction in terms of weak sources of randomness; such a source is a distribution on binary strings that is "random" in the sense of having high "entropy". We will then use an efficient algorithm of Gabizon et al. to extract almost all of the randomness present, obtaining a pseudorandom generator that stretches $O(n)$ bits to $\Omega(n2^n)$ bits.

Keywords:
pseudorandom generator, derandomization, pseudorandomness, oblivious bit-fixing weak sources, extractors, preservation of indistinguishability
To my parents, my sister, and mom Badri.
“Only those who will risk going too far can possibly find out how far one can go.”

— T. S. Eliot
Acknowledgments

Guidance, support and assistance of numerous individuals have made this thesis possible, and there are many to whom I am forever grateful. Mainly, this thesis is based on joint work with my senior supervisor, Dr. Valentine Kabanets. Naturally, I'd like to thank Dr. Kabanets and Dr. Arvind Gupta for introducing me to this field, and for their great input and guidance. I'd like to extend my gratitude to Dr. Gábor Tardos for serving as my examiner on the examining committee. I owe special thanks to my friends Jano Manuch and Peter Zubcsek for their unconditional support and help throughout this thesis. I truly enjoyed my many research meetings with Jano, and am grateful for his patience and meticulous attention to mathematical details. My sincere thanks go to Antonina Kolokolova for her great and timely advice. Most importantly, I owe everything to my parents and sister. They have been with me through it all, and without their love, support and everlasting faith in me I would not have succeeded.
Contents

Approval ii
Abstract iii
Dedication iv
Quotation v
Acknowledgments vi
Contents vii
List of Figures ix

1 Introduction 1
1.1 Blum-Micali-Yao’s pseudorandom generator 4
1.2 Nisan-Wigderson’s Pseudorandom Generator 4
1.3 Distinction between the NW and BMY PRGs 5
1.4 Relationship between hardness and randomness 6
  1.4.1 Hardness implies Randomness 7
  1.4.2 Randomness implies Hardness 7
1.5 Amplification of Hardness 7
  1.5.1 Yao’s XOR Lemma 7
  1.5.2 Amplification from worst-case hardness to Mild hardness 10
  1.5.3 Impagliazzo-Wigderson’s Randomness efficient hardness amplification 10
1.6 Brief introduction to extractors 12
1.7 Sudan-Trevisan-Vadhan Pseudorandom Generator Construction 14
1.8 Significance of our work ........................................... 14

2 Preliminaries ................................................. 18
  2.1 Preliminary definitions ........................................ 18
  2.2 Hardness vs. Indistinguishability .......................... 20
  2.3 Hardness vs. Randomness ..................................... 20
   2.3.1 Randomness implies hardness ............................. 20
   2.3.2 Hardness implies randomness ............................. 22
  2.4 Nisan-Wigderson Pseudorandom Generator Construction . . 24
  2.5 Main Definitions ............................................. 25
   2.5.1 Weak sources and extractors ............................. 25
   2.5.2 Expanders .................................................. 27

3 Our Construction ............................................. 30

4 Preservation of Indistinguishability ......................... 34
  4.1 Preservation of indistinguishability in the cryptography setting .... 35
  4.2 Preservation of indistinguishability in the derandomization setting .... 39

5 Correctness of our construction ................................ 46

6 Conclusion .................................................. 53

A Auxiliary Proofs ........................................... 55
  A.1 Hardness vs. Indistinguishability .......................... 55

Bibliography .................................................. 58
List of Figures

1.1 Derandomization of a randomized algorithm using a pseudorandom generator

\[ G : \{0,1\}^d \rightarrow \{0,1\}^{T(n)} \] .................................................. 3

1.2 Overview of Nisan-Wigderson’s pseudorandom generator [NW94] ................... 5

1.3 Yao’s amplification of hardness using the xor function [Yao82] ................. 9

1.4 Overview of the Impagliazzo-Wigderson Construction [IW97] ................... 11

1.5 Derandomization of a randomized algorithm using an extractor .................. 15

1.6 Sudan-Trevisan-Vadhan’s pseudorandom generator construction using an ex-

   tractor [STV01] ................................................................. 16

1.7 Existing hardness amplification techniques .............................................. 17

2.1 Nisan-Wigderson’s pseudorandom generator [NW94] ............................ 26

2.2 Impagliazzo-Wigderson’s generator used for derandomizing Yao’s XOR lemma

   [IW97] ............................................................................. 28

3.1 An overview of our PRG construction \( G : \{0,1\}^{O(n)} \rightarrow \{0,1\}^{(1-o(1))\delta n2^n} \) ..... 30

3.2 Pseudorandom Generator \( G(z) : \{0,1\}^{O(n)} \rightarrow \{0,1\}^{(1-o(1))\delta n2^n} \) ........... 33

4.1 Preservation of Indistinguishability in different computational settings ....... 36
Chapter 1

Introduction

The problem of improving efficiency of algorithms through more economical use of resources such as time, space and randomness has fascinated computer scientists for decades. For instance, randomness appears to be vital for the efficiency of some algorithms (e.g., polynomial identity testing) and in other cases can be reduced or eliminated (e.g., primality testing). However, there are applications such as cryptography or zero-knowledge proofs where randomness is inherently necessary. Indeed, one of the most fundamental questions in complexity theory is determining the necessity of randomness in efficient algorithms. Namely, is the aesthetic speed up produced by utilizing randomness in algorithm design just due to lack of efficient, advanced deterministic algorithms or does randomness really help?

Resolving this fundamental question is intriguing both philosophically and theoretically. Philosophically, does adding information independent from the input help speed up some computations? Theoretically, elimination of randomness translates into efficient deterministic simulation of randomized complexity classes such as BPP (2-sided probabilistic polynomial time algorithms). Deciphering the role of randomness requires elucidation of the relationship between complexity classes BPP, P (deterministic polynomial time) and EXP (deterministic exponential time). This relationship has been studied from different angles using tools such as pseudo-random generators (referred to as PRGs).

A randomized algorithm $R$, besides its regular input string $x$, needs a secondary input $r$ as the random choices used throughout its computation. Throughout its computation, if a random coin flip is needed, the random bit will be looked up from $r$ and depending on this bit one of two actions takes place. In practice, access to truly random bits is very limited, and more focus has been oriented toward methods of using a short truly random string of
CHAPTER 1. INTRODUCTION

bits (independent coin flips) to produce a longer string that looks random.

Pseudorandom generators are efficient deterministic algorithms which, given a short random seed, provide a longer sequence of bits that are indistinguishable from truly random bits with respect to a well defined observer (e.g., an efficient algorithm with limited resources or circuits of certain size). The goal of a pseudorandom generator is to produce pseudorandom strings that may be used for simulation of randomized algorithms. The idea is that if the observer can not distinguish between the output of the generator and truly random bits, then the generator's pseudorandom output can be used for the observer's internal random bits and the number of truly random bits needed can be reduced.

To simulate a randomized algorithm \( R \) deterministically given a pseudorandom generator \( G \), one can go through all possible input seeds to \( G \), simulate \( R \) on all outputs of the generator, and take a majority vote. This process of using a pseudorandom generator to reduce the number of required random bits takes \( O(2^d(T(G) + T(R))) \) time where \( T(.) \) denotes the running time of the appropriate algorithms and \( d \) is the generator's input length (see Figure 1.1).

Process of reducing (and ultimately eliminating) the number of random bits in a randomized algorithm is referred to as derandomization. Derandomization techniques vary from techniques dependent on the specifics of a randomized algorithm to constructions applicable to any randomized algorithm of a certain complexity class.

As mentioned, the running time of the derandomization process is dominated by the enumeration of seeds to the generator. In particular, a pseudorandom generator that uses a truly random input of logarithmic length makes a deterministic simulation of randomized algorithms with a polynomial overhead possible.

Using the current state-of-the-art techniques, no unconditional explicit construction of pseudorandom generators is known. Informally, constructing such pseudorandom generators entails a tremendous breakthrough in complexity theory as it implies the existence of very hard functions. There are, however, conditional PRG constructions which stretch their inputs exponentially and can provide conditional derandomization of \( \mathsf{BPP} \). In what follows, we present two well known pseudorandom generators whose existences are conditioned on two distinct assumptions.
CHAPTER 1. INTRODUCTION

Randomized algorithm $R(i, r)$ has an error $T(n)$.

Let $n$ be the length of input $i$, and $r$ be a random string of length at most $T(n)$. 

Figure 1.1: Determinization of a randomized algorithm using a pseudorandom generator $G : [0,1]^d \rightarrow [0,1]^{T(n)}$.
CHAPTER 1. INTRODUCTION

1.1 Blum-Micali-Yao’s pseudorandom generator

The pseudorandom generator constructed by Blum, Micali and Yao [BM84, Yao82] (denoted by BMY PRG), uses hardness of a one-way permutation (a length preserving, one-to-one function that can be efficiently computed but is very hard to invert on average) to create pseudorandom bits that fool any probabilistic algorithm running in polynomial time, and hence is a prime candidate for cryptography applications. Hastad, Impagliazzo, Levin and Luby [HILL98] later proved that for the above PRG construction any one-way function suffices. The main idea of the BMY construction is to use a one-way function to produce random bits and argue that if the output is not random with respect to the observer, then the observer can be used to invert the aforementioned one-way function. Since a one-way function, by definition, is computable in polynomial time but can not be inverted in polynomial time, this technique generates pseudorandom bits that are indistinguishable from truly random bits with respect to a distinguisher running in polynomial time.

BMY’s hardness assumption, the existence of a one-way function, is a very strong assumption since it implies \( P \neq \text{NP} \). Nisan and Wigderson in a seminal paper [NW94] introduced a pseudorandom generator that is based on a weaker assumption, and even though has a longer running time, is suitable for derandomizing randomized algorithms. This PRG is based on the assumption that there exists a boolean function computable in exponential time that no circuit of polynomial size can approximate (agree with it on significantly more than half of the inputs). This hardness assumption is weaker than the one in the BMY setting in a sense that the existence of one-way functions implies the validity of the NW assumption but the converse does not necessarily hold.

1.2 Nisan-Wigderson’s Pseudorandom Generator

The intuition behind the Nisan-Wigderson pseudorandom generator (denoted by NW PRG) construction is that if a boolean function \( f \) on \( n \)-bit strings is hard on average (adversaries of certain limited computational resources can not correctly compute \( f \) on slightly more than half the inputs), then applying \( f \) to a random input string should result in an unpredictable bit. Nisan and Wigderson observed that even though applying \( f \) to \( N \) random instances generates \( N \) (computationally) unpredictable bits, this process would be futile for constructing pseudorandom generators. Inherently, a pseudorandom generator stretches its
truly random short seed into a longer (computationally) pseudorandom string but the above process shrinks its input length as it takes \( nN \) random bits and outputs \( N \) bits. Their ingenious idea was to use seeds of small dependence instead of truly random independent ones (see Figure 1.2). They use combinatorial designs to construct several almost independent strings from the short truly random input, and then by applying hard function \( f \) to each such almost independent string produce pseudorandom bits. They prove that if the dependence among the generated strings is small enough, the output will be computationally indistinguishable from a truly random string.

1.3 Distinction between the NW and BMY PRGs

Prior to the Nisan-Wigderson construction of their generator, pseudorandom generators were used in the cryptography setting to stretch their short truly random seed polynomially and fool any randomized polynomial algorithm. However, Nisan and Wigderson focused their construction on the derandomization of particular time classes, and they also permitted
their generator to run in time exponentially in its seed length (i.e., poly($n, 2^l$) where $l$ is the input length to the generator and $n$ is the output length) and fools circuits of size at most its output size (instead of all poly($n$) size circuits).

The $BMY$ PRG can also achieve an exponential stretch; under a much stronger hardness assumption though. Even though both generators can achieve the same hardness vs. randomness trade-offs quantitatively, the $NW$ PRG is constructed under a weaker hardness assumption.*

Another distinction between the two pseudorandom generators is that the existence of $BMY$ PRG implies the existence of one-way functions (which in turn implies $P \neq NP$) while the existence of the $NW$ PRG proves circuits lower bounds for $EXP$. In particular, if $P = NP$, then all functions computable in polynomial time can be inverted in polynomial time as well (i.e., no one-way function can exist), and a polynomial time distinguisher can break the $BMY$ PRG by utilizing the efficient inversion of one-way functions to distinguish the output of the PRG from the uniform distribution. However, if $P = NP$, the $NW$ PRG is provably secure since in this case there will be hard functions which can be used in the construction. Testing whether a string is an output of the $NW$ generator is an $NP$-search problem but even with guessing the seed of the generator correctly, computing the generator on this seed takes more time than the resources of the distinguisher, and so resolving the $P$ vs. $NP$ question, does not necessarily render this generator obsolete.

1.4 Relationship between hardness and randomness

Hardness and randomness are tightly intertwined, and this close interconnection has led to the hardness vs randomness paradigm in which hardness is used to construct pseudorandom generators and PRGs are used to construct hard boolean functions.

This intricate interconnection between hardness and randomness justifies the difficulty encountered in derandomizing randomized algorithms. Namely, one of the reasons that constructing unconditional pseudorandom generators has been so evading is that their constructions would imply existence of explicit hard boolean functions resolving a major open question in complexity theory.

*It should be noted that the $NW$'s hardness assumption, though weaker, is not strictly weaker than the hardness assumption in the $BMY$ setting. In other words, even though the $BMY$ assumption implies the $NW$ assumption, it is unknown whether the $NW$ assumption (existence of hard functions with respect to particular complexity classes) implies the existence of one-way functions.
1.4.1 Hardness implies Randomness

Current PRG constructions can be viewed as non-boolean functions that transform hardness into randomness. Namely, they utilize a hard function to produce pseudorandom bits, and the validity of the construction and hence the integrity of the generated bits depends on the hardness of this function. In both NW and BMY PRGs, a hard function (the notion of hardness differs in these two settings) is used to stretch a short string of truly random bits into a longer pseudorandom string (here again the notion of pseudorandomness is with respect to distinct classes of distinguishers). In Section 2.3.2, we formalize this notion by presenting the parameters of the hard functions that can be derived from these conditional PRGs.

The interconnection between hardness and randomness is even more intricate in that the construction and validity of these generators implies the existence of hard functions. In other words, not only do current PRG constructions require hard functions for their validity, they can be used in demonstrating the other direction of the hardness vs randomness paradigm; randomness implies hardness.

1.4.2 Randomness implies Hardness

Using the BMY pseudorandom generator, one can construct a one-way function. Using a generator whose output is indistinguishable from the uniform distribution for circuits of size \( m \), one can construct a function that has hardness greater than \( m \) (i.e., the smallest circuit that can correctly compute this function has size at least \( m \)). In Section 2.3.1, we prove that both BMY and NW PRGs can be used to construct one-way and computationally-hard functions respectively.

1.5 Amplification of Hardness

1.5.1 Yao's XOR Lemma

Probability theory provides a method to simulate an almost unbiased coin given several biased coins of bounded bias. Recall that the bias of a coin \( C \) is defined as

\[
|\Pr[C \text{ outputs a Head}] - \Pr[C \text{ outputs a Tail}]|.
\]

A coin is considered fair if its bias is zero. In particular, the following experiment illustrates the simulation of a fair coin using \( k \) biased coins of bounded bias \( \delta \).
CHAPTER 1. INTRODUCTION

The XOR technique

Given: Biased coins $\mathcal{G}^1 \ldots \mathcal{G}^k$ that have unknown bias $\alpha_1 \ldots \alpha_k$ respectively
where $\alpha_i \leq \alpha < 1$ for all $1 \leq i \leq k$.

Procedure:

- Flip coins $\mathcal{G}^1 \ldots \mathcal{G}^k$
- Denote the outcome of each $\mathcal{G}^i$ by $c_i$ where
  \[ c_i = \begin{cases} 
  1 & \text{if } \mathcal{G}^i \text{ outputs a Head} \\
  0 & \text{Otherwise} 
  \end{cases} \]
- Xor the outputs together: $b = c_1 \oplus \ldots \oplus c_k$
- Output $b$

Claim: $b$ is an unpredictable bit.

Since the coin flips are independent, the bias of $b$ (denoted by $\mathcal{B}$) is $\alpha_1 \ldots \alpha_k \leq \alpha^k$. The XOR experiment simulates an almost fair coin (a coin whose bias is very close to zero) because probability of $b$ being 1 is $1/2 \pm \alpha^k/2$ and $\alpha^k$ approaches zero exponentially fast (as $k$ increases and approaches infinity in the limit).

Yao in [Yao82] provides a method of amplifying imperfect randomness to (almost) unpredictability. Yao’s technique in the computational complexity setting demonstrates hardness amplification for boolean functions instead of biased random variables described above. It states that the hardness of a boolean function $f$ can be amplified when the outputs of the function on several independent instances are xored together (see Figure 1.3).

Intuitively, if a function is computationally hard, computing the function on several independent instances should be proportionally harder. The xor function has the property that its output value depends on all of its inputs. So, given a somewhat hard function, xoring many evaluations of the function on independent instances ensures that the output resonates the hardness present in each evaluation. Yao’s XOR lemma embodies this intuition and shows that if a function $f$ is (computationally) hard with respect to circuits of a certain size, then slightly smaller circuits can not do much better than a random coin toss when computing $f(x_1) \oplus \ldots \oplus f(x_k)$. 
If a hard boolean function $f$ on $n$-bit strings cannot be computed on at least $\delta$ fraction of its inputs by any circuit of size $s^*$, then the advantage$^1$ of computing such a function is less than $1 - 2\delta$. Intuitively, similar to the XOR technique described above, probability of correctly computing $f(x_1) \oplus \ldots \oplus f(x_k)$ (where $x_i \in U_n$ for all $i$) approaches $\frac{1}{2}$ as $(1 - 2\delta)^k$ diminishes to zero. Yao’s XOR Lemma in the computational setting proves this intuition with one shortcoming, namely it proves amplification of hardness for circuits that are smaller than $s$.

As a generalization of Yao’s XOR lemma, results that amplify hardness through evaluation (without xoring the outputs together necessarily) of a somewhat hard function on many independent instances are called direct product lemmas. Direct product lemmas show that computational difficulty can increase (rather dramatically) by computing a somewhat hard function on many independent instances. In particular, if a function $f$ is hard on average with respect to circuits of size $s$, then circuits of size $s'$ (where $s'$ is smaller than $s$) can not do much better than a random guess when computing $g(x_1, \ldots, x_k) = (f(x_1), \ldots, f(x_k))$ where $x_1, \ldots, x_k$ are $k$ independent random instances.

Yao’s XOR lemma and direct product lemmas are two very powerful tools in complexity theory, but they both have two main drawbacks.

- They demonstrate amplification of hardness while decreasing the size of the distinguishing circuits even though it seems intuitive that if a function is somewhat hard

---

$^*$We say the $f$ can not be computed on $\delta$-fraction of its inputs, if for any circuit $C$ of certain size $s$, $\Pr_{x \in U_n} [C(x) = f(x)] \leq 1 - \delta$. This notion of computability will be further formalized in Section 2.

$^1$Similar to the bias of a coin, computational advantage of a boolean function is defined as the advantage of any circuit of size $s$ in computing the function:

$$|\Pr [C(x) = f(x)] - \Pr [C(x) \neq f(x)]| = |1 - 2 \Pr [C(x) \neq f(x)]|.$$
with respect to circuits of size $s$, then computing the function on many instances and possibly combining the outputs together (e.g., xoring the outputs), should be much harder for circuits of the same size.

- Unfortunately this type of hardness amplification increases the length of the input from $n$ bits to $nk$ bits. In this regard, there have been some developments which show that true independence is not necessary. Impagliazzo and Wigderson [IW97] derandomize Yao's XOR lemma through constructing a generator whose output will be used instead of the $k$ independent instances (Figure 1.4).

1.5.2 Amplification from worst-case hardness to Mild hardness

Babai, Fortnow, Nisan and Wigderson [BFNW93] were the first to amplify the hardness of a function that is very hard in the worst-case (i.e., circuits of a certain size can not compute the function on all instances) to a function that is mildly hard on average (i.e., circuits of a certain size fail to correctly compute this function on at least $\frac{1}{\text{poly}(n)}$ fraction of the inputs). They use error correcting codes to transform the truth table of a boolean function on $n$-bits that small circuits can not correctly compute on at least $\frac{1}{2^n}$ fraction of $n$-bit inputs (a worst-case hard function) to the truth table of another boolean function that small circuits can not correctly compute on at least $\frac{1}{\text{poly}(n)}$ fraction of inputs of length $n$. This construction (conditioned on a hardness assumption) proves that BPP algorithms have subexponential deterministic simulations but is unable (similar to others’ previous attempts) to prove deterministic polynomial time simulation of efficient randomized algorithms (i.e., BPP = P).

To conditionally prove that BPP = P starting from a worst-case hard function, after using BFNW’s construction, in order to utilize the NW PRG, one needs to amplify the mild hardness to get a function that is extremely hard on average. This can be achieved using Yao’s XOR lemma, however, $n^{O(1)}$ random instances are needed to achieve such a hardness amplification. Impagliazzo and Wigderson [IW97] derandomize Yao’s XOR lemma to make such an amplification randomness-efficient enough to prove (conditionally) P = BPP.

1.5.3 Impagliazzo-Wigderson’s Randomness efficient hardness amplification

Impagliazzo in [Imp95] demonstrates how mild hardness can be further amplified to constant hardness (i.e., hardness of a function on which circuits of a certain size fail to correctly
compute this function on at least a constant fraction of the inputs). Impagliazzo and Wigderson [IW97] similar to [BFNW93] use error correcting codes to transform worst-case hardness into mild average-case hardness, and in turn into constant hardness. Then, by derandomizing Yao's XOR Lemma, they transform constant hardness into average-case hardness (which is the level of hardness needed by the NW PRG construction) randomness-efficiently. They construct a generator (see Figure 1.4) which uses its truly random seed of length $O(n)$ to output $n^{O(1)}$ instances that once used in Yao's XOR lemma have the same affect as $n^{O(1)}$ truly random independent instances.

This generator, which derandomizes Yao's XOR lemma, consists of two parts. First, they use combinatorial designs to stretch the input string into many (smaller) strings of small pair-wise dependence. Second, they take advantage of the fact that even a constant-hard function has a subset of instances on which it is essentially unpredictable (proven in [Imp95] and discussed in Section 2.3.2). In particular, the second constituent of their construction is a combinatorial construction that ensures several instances output by the construction (i.e., $B_1 \ldots B_{n^{O(1)}}$ in Figure 1.4) actually fall into the function's hard set.

Later, Sudan, Trevisan and Vadhan [STV01], using an alternative approach (see Figure
1.7) were able to construction a PRG using a mildly-hard on average function. To fully understand their construction, we need to first introduce weak sources of randomness and extractors.

1.6 Brief introduction to extractors

In general, there are distributions that have some randomness but are considered weak sources in a sense that the randomness present is not uniform. An example would be using radioactive decay of Uranium as a random source. Even though such a source may seem random, it can not be proven rigorously that the produced distribution is truly random (i.e., all bits are uniform and independent). For a weak source which has sufficiently high randomness present, the randomness present can be extracted using a randomized, efficient algorithm known as an extractor. An extractor is an efficient algorithm that given a weak source and a short truly random string (acting as a catalyst) outputs a distribution that is statistically close to the uniform distribution. Extractors are very important tools in complexity theory as they can be used for extraction of randomness out of weak sources [NZ96], simulation of randomized algorithms [Zuc96] and construction of PRGs [STV01]. In particular, extractors are very important tools in derandomizing randomized algorithms as they can take on two crucial roles.

- Extractors can extract randomness from a weak random source (such as a Zener diode or last digits of a real time clock), and this extracted randomness can be used as the internal random bits needed by a randomized algorithm. Then, one can derandomize a randomized algorithm \( R \) by enumerating all possible seeds to the extractor (there are \( 2^d \) such strings where \( d \) denotes the length of the extractor's short random seed), applying the extractor, using the output random string as the internal coin flips of \( R \), and taking a majority vote over all outputs of the algorithm (see Figure 1.5).

- They can be used to construct pseudorandom generators useful in derandomizing randomized algorithms. Sudan, Trevisan and Vadhan in [STV01] use an extractor inside the construction of a pseudorandom generator which achieves almost optimal parameters. The significance of using an extractor in this construction (discussed in Section 1.7) is that it (along with the other components of the construction) allows one to construct a PRG without using Yao's XOR lemma for hardness amplification.
CHAPTER 1. INTRODUCTION

Not only can extractors be used for pseudorandom generator constructions such as the construction in [STV01] (discussed in Section 1.7), but also pseudorandom generators can be used to construct extractors. Trevisan [Tre01] demonstrated that any $NW$-type pseudorandom generator that has certain properties can be used for constructing extractors with almost optimal parameters.

One of the earliest and most intuitive constructions of extractors is von Neumann’s construction [vN51] which extracts perfect unbiased bits using biased coin flips. In particular, given a source that with probability $p$ produces a 1 (i.e., a head) and with probability $1 - p$ produces a 0, produce two trials and denote 10 by 1 and 01 by 0, and ignore the remaining two possibilities of 11, 00 by looking at the next pair of bits*. This new distribution is a uniform distribution on \{0, 1\} which with probability $p(1 - p)$ produces a 1 and with probability $p(1 - p)$ a 0.

Von Neumann’s extractor removes the bias from the weak source without requiring any extra random bits, but it has the disadvantage of requiring an indeterminate number of coin flips for extracting and outputting a desired number of random bits. In other words, for extracting $m$ random bits, the number of pairs of coins that need to be flipped until $m$ trials of 01 or 10 are encountered can not be bounded (in the worst case) by a function of $m$.

Extractors that do not require a short additional truly random input are called seedless extractors.¹ Von Neumann’s extractor and the XOR function are examples of such extractors. Von Neumann’s method is a technique for simulating a perfect (unbiased) coin using tosses of a biased coin with unknown but fixed bias. However, in the XOR experiment, it is enough to have an upper bound on the unknown biases of the coins.

Upon a closer look at von Neumann’s intuitive extractor, one notices that the construction is a bit wasteful in a sense that it completely ignores randomness present in the trials resulting in 00 and 11. A construction of Peres [Per92] addresses this issue and, by iteratively going through the information that von Neumann’s construction leaves behind, is able to extract almost (arbitrarily close to) all of the randomness available in the weak source. Namely, given $n$ random initial bits from an imperfect random source, Peres’ construction extracts $nh(p)$ bits (where $h(p) = -p \log p - (1 - p) \log 1 - p$ is the entropy bound of the

*It is assumed that the bits are independent from each other and identically distributed with fixed probabilities.

¹In general, it can be proven that there are distributions from which no extractor can extract even one bit without using a few extra truly random bits as catalyst. So, the majority of extractors receive a very short random seed as secondary input.
weak source) from the weak source.

1.7 Sudan-Trevisan-Vadhan Pseudorandom Generator Construction

Sudan, Trevisan, and Vadhan [STV01] provide two significant techniques for derandomization as alternatives to the previous hardness amplifications which utilized Yao's XOR lemma.

First, they observe that if inside the construction of the NW PRG a mildly hard function is used instead of an extremely hard function, the output distribution will have high entropy instead of being pseudorandom. Their construction (see Figure 1.6) outputs a pseudorandom distribution by using an extractor to extract the randomness present. This construction is significant because it recasts the Impagliazzo-Wigderson construction in the setting of weak sources and extractors and utilizes extractors in the computational instead of the information theoretic setting.

Their second technique is to use polynomial encoding along with list decoding codes to amplify worst-case hardness directly to average-case hardness which can then be used by the Nisan-Wigderson generator (see Figure 1.7). The significant contribution of this technique is that it amplifies hardness without using Yao's XOR lemma.

1.8 Significance of our work

In our construction, we build on the constructions of [STV01] and [IW97]. Sudan, Trevisan and Vadhan apply a mildly-hard function to the construction of Impagliazzo and Wigderson in [IW97] to get a weak source and, using an extractor, extract randomness from this weak source. Impagliazzo and Wigderson by their construction (Figure 1.4) derandomize Yao’s XOR lemma and in essence provide a method of amplifying hardness of a boolean function while maintaining high randomness efficiency. By utilizing the construction of [IW97] and observations in [STV01], we construct a new pseudorandom generator.

Based on an observation from [STV01], we reinterpret the construction of Impagliazzo and Wigderson in [IW97] in terms of weak sources and extractors. We prove that Impagliazzo-Wigderson's construction's layer $W$ (see Figure 1.4) is computationally indistinguishable from a special imperfect random source called an oblivious bit fixing source. We then use
RANDOMIZED ALGORITHM

Randomized algorithm $R(x, r)$ runs in time $T(n)$ where $n$ is the length of input $x$, and $r$ is a random string of length at most $T(n)$.

**Randomized Algorithm**

This extractor takes two inputs, a string of length sampled from the weak source, and a truly random string of length (where $d$ is very small, $d \log n$ in fact). The extractor then outputs a string of length $n$ of which is uniformly distributed.

**Simulating a randomized algorithm using an extractor**

**Derandomization**

Figure 15: Derandomization of a randomized algorithm using an extractor
extractors along with combinatorial designs to output pseudorandom bits.

In our construction, the direct product is not used for amplification of hardness as was the case in the Impagliazzo-Wigderson construction, rather it is utilized for generation of pseudorandom bits. Moreover, similar to Sudan, Trevisan and Vadhan's construction, we use multiple weak sources along with a suitable extractor to derive random bits. However, unlike their construction, we use many such weak sources with small dependence to construct a pseudorandom generator. In other words, we remove complete independence by using combinatorial designs, and prove that indistinguishability remains preserved.
Worst-case Hardness

- Existence of functions in exponential time that cannot be computed by polynomial-size circuits implies deterministic, sub-exponential simulation for infinitely-many functions.

Hard-core Lemma

- Existence of a hard-core for mildly-hard decision functions.

Mild Average-case Hardness

- Pseudorandom Generator
- Expander Walk Generator
- Combinatorial Designs

Our Construction

Figure 1.7: Existing hardness amplification techniques
Chapter 2

Preliminaries

In the following sections, \( x \in_D S \) implies sampling a random variable \( x \) from set \( S \) according to distribution \( D \). If \( S \) is unambiguously apparent, we use \( x \in D \) instead of \( x \in_D S \). The uniform distribution is a distribution in which all elements have equal probability of being chosen. Furthermore, the uniform distribution on \( \{0, 1\}^n \) is denoted by \( U_n \). For conciseness, we denote \( 2^n \) by \( N \). Also, unless otherwise specified, all functions will be boolean functions.

In this section, we provide more details for existing constructions discussed in Section 1, and we start with preliminary definitions and discuss the interconnections between hardness, indistinguishability and randomness.

2.1 Preliminary definitions

Due their elegant representation, circuits are extensively used in complexity theory. It has been shown that efficient algorithms can be simulated by small circuits as well. More precisely, any algorithm running in time \( t \) can be simulated by a circuit of size \( O(t \log t) \).

Definition 1 (Circuit) A circuit is a directed acyclic graph whose nodes can be either an AND, OR, or a NOT gate. The AND and OR gates have fan-in two and the NOT gate has fan-in of one. The size of a circuit is defined as its number of gates.

To fully understand the BMY pseudorandom generator, we need to formally define a one-way permutation. Intuitively, a one-way permutation is a length-preserving, one-to-one function that is easy to compute but very hard to invert.
Definition 2 (One-way Permutation) A permutation \( f : \{0,1\}^n \to \{0,1\}^n \) is a one-way permutation if it can be computed in polynomial time, and furthermore any polynomial-size circuit \( C \) has negligible (smaller than the inverse of any polynomial) probability of inverting \( f \).

Definition 3 (Statistical \( \epsilon \)-indistinguishability) Two distributions \( D_1 \) and \( D_2 \) over the same domain \( \mathcal{D} \), are said to be statistically \( \epsilon \)-indistinguishable if for any event \( E \subseteq \mathcal{D} \),
\[
|\Pr [x \in D_1, E] - \Pr [x \in D_2, E]| \leq \epsilon
\]

Definition 4 (Computational \( \epsilon \)-indistinguishability for size \( t \)) A distribution \( D_1 \) over some domain \( \mathcal{D} \) is said to be \( \epsilon \)-indistinguishable for size \( t \) from another distribution \( D_2 \) (over the same domain) if and only if for all circuits \( C \) of size \( t \),
\[
|\Pr_{x \in D_2}[C(x) = 1] - \Pr_{y \in D_1}[C(y) = 1]| \leq \epsilon
\]
If two distributions are statistically indistinguishable, then they will also be computationally indistinguishable for any size \( t \), but the converse does not necessarily hold.

Definition 5 (\( \epsilon \)-PRG for size \( m \)) A function \( G : \{0,1\}^l \to \{0,1\}^{m(l)} \) is an \( \epsilon \)-PRG if \( G(x) \) is \( \epsilon \)-indistinguishable from \( U_{m(l)} \) for size \( m \).

Definition 6 (Bias) Bias of a 0-1 random variable \( X \) is defined as the absolute value of the difference between its probability of success and failure:
\[
\text{Bias}[X] = |\Pr [X = 0] - \Pr [X = 1]|
\]
Analogously, bias of a boolean function \( f : \{0,1\}^n \to \{0,1\} \) is defined as:
\[
\text{Bias}[f] = |\Pr_{x \in U_n}[f(x) = 0] - \Pr_{x \in U_n}[f(x) = 1]|
\]

Definition 7 (Computational hardness for boolean functions) A boolean function \( f : \{0,1\}^n \to \{0,1\} \) is \( \delta \)-hard\(^*\) with respect to circuits \( C \) of size \( t \) if for any such circuit the following holds:
\[
\Pr_{x \in U_n}[C(x) \neq f(x)] \geq \delta
\]
Depending on the value of \( \delta \), hardness of function \( f \) will be referred to as:
\(^*\)For boolean functions, \( \delta \) can be at most \( \frac{1}{2} \) as a boolean function can be guessed correctly with probability \( \frac{1}{2} \).
CHAPTER 2. PRELIMINARIES

- **Worst-case hardness**: \(0 < \delta \leq \frac{1}{2^n} \)
- **Mild (Average-case) hardness**: \(\delta = \frac{1}{p(n)}\) for some polynomial \(p(n)\)
- **Constant (Average-case) hardness**: \(\delta = \theta(1)\)
- **(Extreme) Average-case hardness**: \(\delta = \frac{1}{2} - \frac{1}{2^{\Omega(n)}}\)

Average-case hardness presents a better portrayal of hardness and intractability of problems in practice as it highlights the performance of an algorithm on average. Furthermore, it is very significant in complexity theory and derandomization as it makes construction of pseudorandom generators possible. However, its inherent difficulty has inclined researchers to start from functions that are only hard in the worst case and amplify this hardness to average-case hardness.

**Definition 8 (Computational Inapproximability)** A boolean function \(f\) on \(n\)-bits can not be approximated by circuits of size \(s(n)\) if \(f\) is \((\frac{1}{2} - \frac{1}{s(n)})\)-hard with respect to circuits of size \(s(n)\).

### 2.2 Hardness vs. Indistinguishability

Yao in [Yao82] observes that hardness and indistinguishability are equivalent.

**Lemma 9 ([Yao82])** For boolean function \(f : \{0,1\}^n \rightarrow \{0,1\}\) and \(x \in U_n\), \(xf(x)\) is \(\epsilon\)-indistinguishable for size \(s\) from \(y \in U_{n+1}\) if and only if \(f\) is \((\frac{1}{2} - \epsilon)\)-hard for size \(s + \theta(1)\).

This lemma, proven in Appendix A, is an essential ingredient in the proof of the correctness of NW-style PRGs.

### 2.3 Hardness vs. Randomness

#### 2.3.1 Randomness implies hardness

Computational randomness is based on the notion of indistinguishability with respect to a particular observer. Moreover, as discussed in Section 2.2, indistinguishability directly

\(^1\)Even one wrong instance suffices in this case.
corresponds to hardness. The following theorems demonstrate that the existence of a pseudorandom generator whose output is computationally indistinguishable from the uniform distribution can be used to construct a hard function. Depending on the pseudorandom generator used, different hard boolean functions can be constructed. Following this abstraction, one can show that the existence of a pseudorandom generator that, similar to the BMY PRG, runs in polynomial time and fools all polynomial time randomized algorithms implies the existence of a one-way function*.

**Theorem 10** If there is a polynomial time pseudorandom generator $G : \{0,1\}^n \rightarrow \{0,1\}^{2n}$ that fools all polynomial size circuits, then there is a corresponding explicit one-way function $f : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ where $f(xy) = G(x)$.

Also, one can prove that the existence of a pseudorandom generator that, similar to the NW PRG, runs in exponential (in its seed length) time and fools a restricted class of distinguishers implies the existence of a computationally hard function.

**Theorem 11** ([ISW99]) If for some non-negative constant $\epsilon$ smaller than $\frac{1}{2}$, there is an $\epsilon$-PRG $G_I : \{0,1\}^l \rightarrow \{0,1\}^{n(l)}$ for size $n(l)$ running in time $2^l$, then there is a $2^{O(l)}$-time computable boolean function $f_I : \{0,1\}^l \rightarrow \{0,1\}$ with $S_{f_I} > n(l - 1)$ where $S_{f_I}$ denotes the size of the smallest circuit that correctly computes $f_I$.

**Proof:** Define a boolean function $f_I : \{0,1\}^l \rightarrow \{0,1\}$ as:

$$f_I(x) = \begin{cases} 1 & \exists z \in \{0,1\}^{n(l-1)-l} \text{ such that } xz \text{ is an output of } G_{l-1} \\ 0 & \text{otherwise} \end{cases}$$

Suppose $S_{f} \leq n(l - 1)$, i.e., the smallest circuit $C$ correctly computing $f$ on all inputs has size at most $n(l - 1)$. We derive a contradiction by showing that this circuit distinguishes the output $G_{l-1}$ from $U_{n(l-1)}$ with probability more than $\frac{1}{2}$.

$f$ outputs 1 only on prefixes of the outputs of the generator:

$$\Pr_{y \in U_{l-1}} [C((G_{l-1}(y))) = 1] = 1,$$

and there are at most $2^{l-1}$ such possible prefixes because the seed length of $G_{l-1}$ is $l - 1$. So,

* A proof of this theorem appears in Appendix A.
Thus, $\Pr_{x \in U_n} [C(x) = 1] \leq \frac{2^{l-1}}{2^{n(l-1)}} \leq \frac{1}{2}$. This contradicts the $\epsilon$-indistinguishability of $G_{l-1}$, and so it must be that $S(f_l) > n(l-1)$.

Furthermore, $f_l$ can be correctly computed by enumerating all seeds $y$ to $G_{l-1}$ and computing $G_{l-1}(y)$ since $f_l$ outputs 1 on prefixes of the output of the generator and 0 otherwise. Thus, $f_l$ can be computed in time $2^{O(l)}$. 

### 2.3.2 Hardness implies randomness

**Hard-bit from a one-way permutation:**

Goldreich and Levin [GL89] prove that from a very hard function (such as a one-way permutation) one can derive an unpredictable bit, even when given $f(x)$ and $r$. Intuitively, since a one-way permutation $f(x)$ is hard to invert, the inner product of $x$ and a random string $r$ (where $|x| = |r|$) should be an (almost) unpredictable bit.

Denote the inner product in $\mathbb{GF}(2)$ of two $n$-bit strings $x$ and $y$ by: $(x, y) = \sum_{i=1}^{n} x_i y_i \mod 2$.

The Goldreich-Levin method of producing an unpredictable bit from a one-way permutation can be formalized as the following theorem.

**Theorem 12 (Goldreich-Levin's hard-bit Theorem [GL89])** Suppose $f_n : \{0, 1\}^n \to \{0, 1\}^n$ is a (polynomial-time) one-way permutation such that the following holds for all algorithms $A$ running in time $s(n)$:

$$\Pr_{x \in U_n} [A(x) = f_n^{-1}(x)] \leq \frac{1}{s(n)}$$

Then, for all algorithms $A'$ running in time $s^{1/2}(n)$, the following holds:

$$\Pr_{r, x \in U_n} [A'(f_n(x), r) = (x, r)] \leq \frac{1}{2} + O\left(\frac{1}{s(n)}\right)$$

The Goldreich-Levin hard-bit theorem is very significant because it provides a technique of deriving an almost unpredictable bit from a one-way permutation. This can be in turn used to construct a pseudorandom generator that stretches its seed by one.

**Theorem 13 (PRG with a one-bit stretch [BM84])** Let $f$ be a one-way permutation on $n$ bits that no circuit of size at most $s(n)$ can invert it with probability greater than $\frac{1}{s(n)}$. Then, $G : \{0, 1\}^{2n} \to \{0, 1\}^{2n+1}$ defined as $G(x, r) = r f(x) b(x, r)$ (where $b(x, r) = (x, r)$ is the hard-bit produced in Theorem 12) is an $\frac{1}{s(n)}$-PRG for size $s(n)$. 

Intuitively, using a random input $x$ and $r$, the first $2n$ bits of $G$ can not be predicted because $f$ is a permutation and its input is a random string. Also, because $b(x, r)$ is a hard bit of $f$ with respect to circuits of size $s(n)$, the last bit of $G$ will be almost unpredictable for circuits of size $s(n)$. The shortcoming of this pseudorandom generator is that it only stretches its input length by one. This can be remedied by the following theorem of Goldreich that demonstrates that the existence of a pseudorandom generator with stretch of one bit makes the construction of a pseudorandom generator with a polynomially long stretch possible.

**Theorem 14** ([GM84a]) *Given a pseudorandom generator $G_1 : \{0,1\}^n \rightarrow \{0,1\}^{n+1}$, there exists a pseudorandom generator $G_{l(n)} : \{0,1\}^n \rightarrow \{0,1\}^{l(n)}$ for every polynomial $l(n)$. $G_{l(n)}$ applies $G_1$ to a random seed $x$, records the last bit of $G_1(x)$ and then applies $G_1$ to the remaining portion. This procedure is repeated for $l(n)$ steps and outputs the recorded $l(n)$ (pseudorandom) bits.*

**Hard-core set in the computational setting:**

The notion of deriving an unpredictable bit from a very hard function can be further generalized in the computational setting to that of a hard-core set.

**Definition 15 (Hard-core set)** $M$ is an $\epsilon$-hard-core set of boolean function $f_n$ for size $g$ if for all circuits $C$ of size $g$,

$$\Pr_{x \in U} [C(x) = f_n(x)] < \frac{1}{2} + \frac{\epsilon}{2}$$

By the above definition, any function that has a sufficiently large hard-core set is also weakly hard.

**Observation 16** If a function has a hard-core set of size $\delta 2^n$, then the function must be $\delta (\frac{1}{2} - \frac{\epsilon}{2})$-hard.

**Proof:** Let $|M| = \delta 2^n$ (i.e., $\delta$-fraction of instances of $f$ form a hard-core set), and note that all probabilities are taken over $x \in U_n$.

$$\Pr[C(x) = f(x)] = \Pr[C(x) = f(x) \& x \in M] + \Pr[C(x) = f(x) \& x \notin M]$$

$$= \Pr[x \in M] \Pr[C(x) = f(x)|x \in M] + \Pr[x \notin M] \Pr[C(x) = f(x)|x \notin M]$$

$$\leq \delta (\frac{1}{2} + \frac{\epsilon}{2}) + (1 - \delta) 1^1 = 1 - \delta (\frac{1}{2} - \frac{\epsilon}{2})$$

\footnote{since any probability is upper bounded by one}
This proves that the existence and size of a hard-core set of a boolean function directly correlates with its hardness. Impagliazzo’s Hard-core lemma proves that the converse holds as well. Namely, he proves that every weakly-hard function has a hard-core set of substantial size.

**Lemma 17 (Impagliazzo’s Hard-core Lemma [Imp95])** Suppose no circuit of size $s$ can compute $P : \{0,1\}^l \rightarrow \{0,1\}$ on more than $1 - \delta$ fraction of the inputs in $\{0,1\}^l$. Then, for every $\epsilon > 0$, there exists an $\epsilon$-hardcore set $H \subseteq \{0,1\}^l$ such that $|H| \geq \delta 2^l$ and $P$ is $\epsilon$-hardcore on $H$ for size $s' = \Omega(\epsilon^2 \delta^2 s)$.

This lemma guarantees that every $\delta$-hard (with respect to circuits of size $g$) boolean function on $n$-bit strings is $\epsilon$-hardcore on at least $\delta$ fraction of its inputs with respect to circuits of size $g' = \Omega(\epsilon^2 \delta^2 g)$.

### 2.4 Nisan-Wigderson Pseudorandom Generator Construction

Inspired by Yao’s amplification of hardness and direct product lemmas, the first intuition for constructing a pseudorandom generator would be to use a $(\frac{1}{2} - \frac{1}{\log n})$-hard boolean function $f$ to construct the following almost-unpredictable function:

$$G(x_1, \ldots, x_k) = f(x_1) \cdots f(x_k)$$

where all $x_i \in U_n$.

The problem with this construction is that it outputs a shorter string than its seed. A resolution to this shortcoming is to evaluate the hard function on almost independent strings. Nisan and Wigderson evaluate an average-case hard function on almost independent strings, and prove that the resulting bits are (computationally) pseudorandom (see Figure 2.1). Nisan and Wigderson, who first used such combinatorial structures in the context of pseudorandom generators, labeled them as combinatorial designs. Such combinatorial designs exist and have been long studied in combinatorics as packings.

**Definition 18 (Packings or Combinatorial Designs)** A collection of sets $\{S_1, \ldots, S_n\}$ where $S_i \subseteq \{1, \ldots, d\}$ is a $(l, m)$-design if

- $\forall i, |S_i| = m$
- $\forall i \neq j, |S_i \cap S_j| \leq l$

By the probabilistic method, one can prove the existence of such designs, and by brute force, one can construct such sets one after the other, while preserving their properties.
Lemma 19 ([NW94]) For all integers \( n \), there exists a \((\log n, C \log n)\)-design \( \{ S_1, \ldots, S_n \} \), where \( S_i \subset \{ 1 \ldots d \} \) and \( d = O(C^2 \log n) \). Moreover, the design can be computed by a Turing machine running in time polynomial in \( n \).

Let \( X|_S \) denote substring of \( X \) that are indexed by the elements of \( S \).

Definition 20 Given a \((1,m)\)-design \( \{ S_1, \ldots, S_n \} \) with each \( S_i \subseteq \{ 1, \ldots, d \} \) and average-case hard function \( f : \{0,1\}^d \to \{0,1\} \), define \( NW : \{0,1\}^d \to \{\{0,1\}^d\}^n \) to be
\[
NW(x) = (x|_{S_1} \ldots x|_{S_n})
\]

Furthermore, denote the Nisan-Wigderson generator based on \( f \) by:
\[
f(NW(x)) = f(x|_{S_1}) \ldots f(x|_{S_n})
\]

The main result of Nisan and Wigderson’s pseudorandom generator construction (see Figure 2.1) can be formulated as the following theorem:

Theorem 21 ([NW94]) For every function \( s \), \( d \leq s(d) \leq 2^d \), the following are equivalent:

- For some \( c > 0 \), some function computable in deterministic exponential time can not be approximated by circuits of size \( s(d^c) \).
- For some \( c > 0 \) there exists a \((\frac{1}{s(d^c)})\)-pseudorandom generator \( f(NW) : \{0,1\}^d \to \{0,1\}^{s(d^c)} \) which runs in time exponential in \( d \).

Thus, a function that can not be approximated by polynomial size circuits can be used to build a pseudorandom generator \( f(NW) : \{0,1\}^{O(\log n)} \to \{0,1\}^n \). Enumerating over all possible seeds of such a pseudorandom generator, using its output as the random bits used in any polynomial randomized algorithm, and outputting a majority vote, leads to efficient derandomization of randomized polynomial algorithms; \( \text{BPP} = \text{P} \).

2.5 Main Definitions

2.5.1 Weak sources and extractors

Definition 22 (min-entropy) The min-entropy of a distribution \( X \) is:
\[
H_\infty = \min_x \{-\log \Pr [X = x]\}
\]
Min-entropy is an extremely important concept because it is an indicator of the worst-case randomness present in a distribution just as Shannon’s entropy (generally known as entropy) represents the average-case randomness in a distribution.

**Definition 23 (Extractors)** A $(k, \varepsilon)$-extractor is a function $\text{EXT}: \{0, 1\}^n \times \{0, 1\}^d \rightarrow \{0, 1\}^m$ such that for every distribution $X$ with min-entropy at least $k$, the distribution $\text{EXT}(X, U_d)$ is $\varepsilon$-close to $U_m$.

An $(n, k)$-oblivious bit-fixing source is another example of a weak source which can be said to have an embedding of the uniform distribution. In particular, the probability of choosing any string $x \in \{0, 1\}^n$ is at most $\frac{1}{2^k}$ (i.e., this distribution has min-entropy at least $k$).

**Definition 24 (Oblivious Bit-Fixing Sources)** A distribution $X$ over $\{0, 1\}^n$ is an Oblivious $(n, k)$-bit-fixing source if there is a subset $S = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ such that $X_{i_1}, X_{i_2}, \ldots, X_{i_k}$ is uniformly distributed over $\{0, 1\}^k$ and for any $h \notin S$, $X_h$ is constant.

*Another type of a bit-fixing source would be a non-oblivious bit-fixing source in which the bits outside of $S$ may depend on the bits in $S$.*
Definition 25 (Convex Combination of Distributions) A distribution $D$ is said to be a convex combination of distributions $D_1, D_2, \ldots, D_t$ if there is a distribution $I$ on $\{1, \ldots, t\}$ such that $D$ can be realized by choosing $i \in \{1, \ldots, t\}$ according to distribution $I$ and taking a sample $x$ from $S_i$.

Min-entropy will be preserved under convex combination. Namely, the convex combination of $n$ weak sources $W_1, W_2, \ldots, W_n$ of respective min-entropy $k_1, k_2, \ldots, k_n$ has min-entropy

$$k = \min\{k_1, k_2, \ldots, k_n\}:$$

where $p_1 + p_2 + \ldots + p_n = 1$ and each $p_i$ denotes the probability of choosing weak source $W_i$.

Definition 26 (Oblivious bit-fixing extractor) A function $E : \{0,1\}^n \rightarrow \{0,1\}^m$ is a deterministic $(k, \epsilon)$ bit-fixing source extractor if $E(X)$ is statistically $\epsilon$-close to $U_m$ for any oblivious $(n, k)$-bit-fixing source $X$.

Theorem 27 ([GRSO4]) For every constant $\gamma > 0$ and for any $k$, there is an explicit* deterministic $(k, \epsilon)$-bit-fixing source extractor $E : \{0,1\}^n \rightarrow \{0,1\}^m$ where $m = k - n^{\frac{1}{2} + \gamma}$ and $\epsilon = 2^{-\Omega(n^\gamma)}$.

2.5.2 Expanders

An expander graph is an important combinatorial tool in theoretical computer science which has seemingly contradicting properties. Namely, an expander graph on one hand is a sparse graph with a relatively small vertex degree, and on the other hand, it is extremely well connected.

Definition 28 (Expanders) A $(k, \alpha)$-expander graph $G = (V, E)$ is a graph on $n$ vertices where any vertex has at most $d$ neighbors (where $d$ is a small constant (preferably independent of $n$)), and every subset $S$ of the vertices of size $|S| \leq k$ has at least $\alpha |S|$ neighbors for some constant $\alpha$.

In particular, starting from any arbitrary vertex in the expander graph, and making a random walk on the graph by choosing the next vertex uniformly at random out of the neighbors of the current vertex ensures that the probability of stagnation on a subset of the vertices is exponentially small (after sufficiently many steps of the random walk).

* An extractor is said to be explicit if it is computable in polynomial time.
Definition 29 (Expander Walk Generator \cite{IW97}) Fix an explicit expander graph \(G\) on vertex set \(\{0,1\}^n\), of constant degree 16 and small second eigenvalue (say \(< 8\)). Let the expander walk generator \(EW : \{0,1\}^n \times [16]^{k-1} \rightarrow ([0,1]^n)^k\) be defined by \(EW(v;d) = (v_1, v_2, \ldots, v_k)\) where \(v_1 = v\) and \(v_{i+1}\) is the \(d_i\)th neighbor of \(v_i\) in \(G\).

Definition 30 (\cite{IW97}) A pseudorandom generator \(G\) producing a distribution \(x_1, \ldots, x_k\) on \(([0,1]^n)^k\) is \((k', q, \delta)\)-hitting if for any sets \(H_1, \ldots, H_k \subseteq \{0,1\}^n\), \(|H_i| \geq \delta 2^n\) we have \(\Pr[\{|i| x_i \in H_i\}] < k' \leq q\).

Theorem 31 (\cite{CW89}, \cite{IZ89}) For all \(k \leq n\), \(EW\) from Definition 29 is \((\frac{k}{10}, 2^{-\frac{k}{10}}, \frac{1}{2})\)-hitting.
Corollary 32  Let $EW : \{0, 1\}^n \times [16]^{n-1} \to \{0, 1\}^{n^2}$ be the generator defined in Definition 29 where $EW(y) = (y_1, \ldots, y_n)$. Then, for any set $H$ of size at least $\frac{2n}{3}$,
$$\Pr \left[ |\{i | y_i \in H\}| < \frac{n}{10} \right] \leq 2^{-\frac{n}{10}}.$$  

This corollary will be useful in the context of hard-core sets of predicates for our construction outlined in Section 3. Given a constant-hard boolean function $f$ with its corresponding hard-core set $H$, evaluating this function on the output of $EW$ ensures that, almost certainly, more than $\frac{n}{10}$ of them will fall into the hard-core set of $f$, and therefore evaluation of $f$ on them will be (almost) unpredictable.

Using the properties of $EW$ and combinatorial designs, Impagliazzo and Wigderson in [IW97] construct a generator used for derandomizing Yao's XOR Lemma. Layer $W$ of their construction (see Figure 2.2) can be interpreted as a weak source with high min-entropy. They extract a random bit from this weak source by applying the XOR function. However, by utilizing a better extractor, one can extract more random bits from this weak source of high min-entropy.
Chapter 3

Our Construction

In this section, we discuss our main construction of a pseudorandom generator $G(z)$ which will consist of three layers (see Figure 3.1). The goal of this construction is to construct a pseudorandom generator that stretches the short, truly random input $z$ into many (computationally) random bits. Our construction, which is in many ways based on [STV01], consists of three layers that are pipelined (i.e., the output of a preceding layer is used as input for the next). Inspired by [STV01], we utilize a special type of extractors for our construction. Our construction takes as input a truly random string $z$ of length $O(n)$ and outputs a pseudorandom string of length $(1 - o(1))\delta n^{2n}$ for some positive constant $\delta$ smaller than one.

Figure 3.1: An overview of our PRG construction $G : \{0, 1\}^{O(n)} \rightarrow \{0, 1\}^{(1-o(1))\delta n^{2n}}$

In what follows, we introduce the the three layers of our construction.
1. Combinatorial designs: \[NW^\text{long}\]

Using combinatorial designs (recall Definition 18), expand \(c_n\) truly random bits into \(2^n = N\) strings of length \(c_1 n\) where \(c_1 < c\). Furthermore, any two such strings have at most \(\gamma_1 n\) bits in common (where \(\gamma_1\) is some positive constant: \(\gamma_1 < c_1 < c\)).

In our construction, \(NW^\text{long} : \{0,1\}^{c_n} \to (\{0,1\}^{c_1 n})^N\) can be formulated as:

\[NW^\text{long}(z) = (z_1, z_2, \ldots, z_N)\] such that:

- \(\forall i \; |z_i| = c_1 n\)
- \(\forall i \neq j, |z_i \cap z_j| \leq \gamma_1 n\)

2. Combinatorial designs and an expander walk generator: \[P(NW^\text{short} \oplus EW)\]

The second layer consists of a predicate \(P\) and two components denoted by \(NW^\text{short}\), and \(EW\).

- **Predicate \(P\)** is a \(\delta\)-hard boolean function (for some constant \(\delta \in (0, \frac{1}{2})\)) with respect to circuits of size \(s\) and is applied to the output of \(NW^\text{short} \oplus EW\).
- **\(NW^\text{short}\)** uses combinatorial designs to stretch its input seed into \(n\) strings of length \(c_2 n\) with \(\log n\) pairwise intersections.
- **Generator \(EW\)** is a random walk on an expander as defined in Definition 29. This generator ensures that many of the instances belong to a subset of instances (namely the hard-core set of \(P\)) on which the application of \(P\) is unpredictable (with respect to specific distinguishers such as circuits of a certain size). This enables us to alleviate the need for complete independence since if the hard-core set of predicate \(P\) is large enough, many generated instances will be computationally unpredictable.

Let \((z^1, \ldots, z^i, \ldots, z^N) = NW^\text{long}(z)_{z \in U_{cn}}\). The second layer forms sources denoted by \(Y_1 \ldots Y_N\) (see Figure 3.2) where each \(Y_i\) is formally defined as:

**Definition 33** Each source \(Y_i\) is defined as:

\(Y_i = Y(z^i)\) where \(Y : \{0,1\}^{c_1 n} \to \{0,1\}^n\) and

\(Y(z^i) = (P(X^i_1 \oplus Y^i_1) \ldots P(X^i_n \oplus Y^i_n))\) given that
3. Oblivious bit-fixing extractor EXT:

The third layer is a collection of special extractors applied to the outputs of each source $T_i$ in the second layer.

The extractors used in this layer are oblivious bit-fixing extractors designed for bit-fixing sources in [GRS04]. This extractor construction [GRS04] holds for $(n, k)$ oblivious bit-fixing sources where $k \gg \sqrt{n}$. This requirement is satisfied in our construction since $O(\delta n) \gg \sqrt{n}$ for large enough $n$. The extractor extracts $m = (1 - o(1))k$ bits from a $(n, k)$-oblivious bit-fixing source. Utilizing this extractor in our construction $m = (1 - o(1))O(\delta n)$ bits can be extracted from each source, and our generator will have an output of size $N(1 - o(1))O(\delta n)$.

The first two layers both use combinatorial designs, albeit with distinct properties. In the $NW^{long}$ layer, an input $z$ is stretched into $N = 2^n$ substrings $z^1 \ldots z^N$. In the second layer, each input seed is stretched into $n$ substrings of length $O(n)$. The purpose of the first layer is to decrease the number of truly random bits needed for generating $N$ sources. Namely, the combinatorial designs used in the first layer make the generation of $T_1 \ldots T_N$ possible using only $O(n)$ instead of $O(n)N = 2^{O(n)}$ truly random bits.

Since predicate $P$ is a $\delta$-hard predicate with a hard-core set of size at least $\delta 2^n$, incorporating $EW$ in our construction ensures that $P(NW^{short} \oplus EW)$ has at least $\frac{n}{10} = O(\delta n)$ random bits almost certainly (i.e., with probability exponentially close to 1). So, amount of true randomness present in each source $T_i$ is at least $O(\delta n)$ bits. In particular, we prove that each $T_i$ source in the second layer is computationally indistinguishable from a convex combination of oblivious bit fixing sources, and is suitable for the application of oblivious-bit extractors.
Figure 3.2: Pseudorandom Generator $G(z) : \{0, 1\}^{O(n)} \rightarrow \{0, 1\}^{(1-o(1)) \delta n 2^n}$
Chapter 4

Preservation of Indistinguishability

The correctness of our construction relies on proving:

1. each $T_i$ source in the second layer (see Figure 3.2) is computationally indistinguishable from a convex combination of oblivious bit fixing sources with high-enough min-entropy for the OBFS extractor of [GRS04] to be applied, and that

2. using designs in the first layer does not deteriorate the quality of the sources.

Each output source $T_i$ will have high min-entropy because for any string $x$ that belongs to a hard-core set of $P$, $P(x)$ will be almost computationally unpredictable, and each source is constructed to contain many such unpredictable instances. Obviously, the number of such unpredictable bits is bounded by the size of the predicate's hard-core set. By Impagliazzo's Hard-core Lemma (Lemma 17), and properties of $EW$, there are at least $\frac{3 \delta n}{10} = O(\delta n)$ such unpredictable bits present in each source, and using an appropriate extractor, one should be able to extractor almost all of the randomness present.

Using designs in the first layer can be viewed as taking multiple samples of $\text{EXT}(P(NW^{\text{short}} \oplus EW))$. Taking multiple samples of a distribution has been long studied in complexity theory. In fact, an important property of a $BMY$-style PRG is that its indistinguishability property stays preserved under multiple samplings. In other words, if the output of the pseudorandom generator is indistinguishable from the uniform distribution with respect to a specific adversary, taking more than one sample from the generator will not assist the adversary in distinguishing the generator from the uniform distribution. So, proving that using designs in the first layer of our construction preserve indistinguishability
between the output and the uniform distribution is particularly interesting since it reiterates on this important property of pseudorandom generators.

In what follows, we discuss how indistinguishability is preserved under multiple samples. We prove the preservation of indistinguishability in both cryptography and derandomization setting. We further show that indistinguishability with respect to (nonuniform) circuits is always preserved even though the same does not hold with respect to uniform algorithms in which case a further condition (efficient samplability of the two distributions) is required for the preservation of indistinguishability.

We show that if two distributions $X$ and $Y$ on $n$-bit strings are computationally indistinguishable (indistinguishability with respect to certain observers), then taking $k$ independent samples from each distribution does not jeopardize indistinguishability. We will show the preservation of indistinguishability in both cryptography and derandomization settings, and in both settings we consider uniform and non-uniform distinguishers (see Figure 4.1). This enables us to further illuminate the distinctions between the cryptography and derandomization settings as well as emphasize the requirements for preservation of indistinguishability. The proof of preservation of indistinguishability follows a standard argument called the hybrid argument [GM84b].

### 4.1 Preservation of indistinguishability in the cryptography setting

In the context of cryptography, efficiency (namely running in polynomial time) is crucial, and so it only makes sense that indistinguishability is preserved for \textit{polynomially} many samples of two initially indistinguishable distributions (i.e., $k = \text{poly}(n)$). In this case, since any polynomial time algorithm can be used as a distinguisher, the amount of information given out should be restricted.

Imagine that distribution $X$ is the output of a pseudorandom generator (e.g., the \textit{BMY generator}) and distribution $Y$ is the uniform distribution. In this case, since the generator runs in polynomial time, providing more information such as the input seed to the generator eliminates indistinguishability completely. The distinguisher and the generator both run in polynomial time. If the seed to the generator, $\sigma$, is revealed, a polynomial-time distinguisher can compute $G(\sigma)$ and distinguish the output of the generator from the uniform distribution. Thus, in the cryptography setting, indistinguishability can only be preserved
CHAPTER 4. PRESERVATION OF INDISTINGUISHABILITY

Preservation of indistinguishability
under different computational settings

Cryptography

Complete independence
Non-Uniform
Uniform

Derandomization

Complete independence
Non-Uniform
Uniform
Small dependence
Non-Uniform
Uniform

The dotted lines indicate that indistinguishability will not be preserved in that particular setting.

Figure 4.1: Preservation of Indistinguishability in different computational settings
for polynomially many independent samples where no extra information is given out. As we will see later on, the analogous situation does not hold in the derandomization setting since giving some extra information (such as the input seed in the case of indistinguishability between a pseudorandom generator and the uniform distribution) is not only harmless but also necessary in some cases. In a sense, preservation of indistinguishability is more significant in the latter case because here preservation is maintained despite the higher power and adversary’s extra information.

Given two distributions $X$ and $Y$ (both on $n$-bits), denote the $k$ independent samples from $X$ and $Y$ respectively by $X_1 \ldots X_k$ and $Y_1 \ldots Y_k$.

**Theorem 34** If distributions $X$ and $Y$ are $\epsilon$-indistinguishable in the cryptography setting for distinguishers of (at most) polynomial size, then $X_1 \ldots X_k$ will be $\epsilon k$-indistinguishable from $Y_1 \ldots Y_k$ for polynomial size distinguishers.

**Proof Sketch:** We assume that there is a distinguisher $D$ distinguishing $X_1 \ldots X_k$ from $Y_1 \ldots Y_k$ with probability greater than $\epsilon k$ and derive a contradiction to $\epsilon$-indistinguishability of $X$ and $Y$.

We construct $k$ intermediate distributions (called hybrids) that sample from both $X$ and $Y$, and then using the properties of non-uniformity (in the case of non-uniform distinguishers) and efficient samplability (in the case of uniform distinguishers) we derive a contradiction.

**Proof:** Assume that there is a distinguisher $D$ which distinguishes $X_1 \ldots X_k$ from $Y_1 \ldots Y_k$ with probability greater than $\epsilon k$:

$$| \Pr[D(X_1 \ldots X_k) = 1] - \Pr[D(Y_1 \ldots Y_k) = 1] | > \epsilon k$$

Without loss of generality we can assume $\Pr[D(X_1 \ldots X_k) = 1] > \Pr[D(Y_1 \ldots Y_k) = 1]$ to get:

$$\Pr[D(X_1 \ldots X_k) = 1] - \Pr[D(Y_1 \ldots Y_k) = 1] > \epsilon k$$

Build hybrids $H_0 \ldots H_k$ where

*However, the size of the new distinguisher will be a smaller polynomial.*
CHAPTER 4. PRESERVATION OF INDISTINGUISHABILITY

\[ H_0 = Y_1 \ldots Y_k \]
\[ H_1 = X_1 Y_2 \ldots Y_k \]
\[ \vdots \]
\[ H_i = X_1 \ldots X_i Y_{i+1} \ldots Y_k \]
\[ \vdots \]
\[ H_k = X_1 \ldots X_k. \]

Success probability of distinguisher \( D \) (inequality (4.1)) can be rewritten in terms of the two extreme hybrids \( H_0 \) and \( H_k \):

\[
\Pr [D(H_k) = 1] - \Pr [D(H_0) = 1] > \epsilon k
\]
\[
\Pr [D(H_k) = 1] - \Pr [D(H_{k-1}) = 1] + \Pr [D(H_{k-1}) = 1] - \ldots - \Pr [D(H_1) = 1] +
\]
\[
\Pr [D(H_1) = 1] - \Pr [D(H_0) = 1] > \epsilon k
\]
\[
\sum_{i=1}^{k} (\Pr [D(H_i) = 1] - \Pr [D(H_{i-1}) = 1]) > \epsilon k
\]

Hence,

\[
\exists i, \Pr [D(H_i) = 1] - \Pr [D(H_{i-1}) = 1] > \frac{\epsilon k}{k} = \epsilon^* \tag{4.2}
\]

We build a new distinguisher that using \( D \) distinguishes \( X \) from \( Y \) and hence yields a contradiction to the computational indistinguishability of \( X \) and \( Y \).

\[
\Pr [D(X_1 \ldots X_{i-1}X_iY_{i+1} \ldots Y_k) = 1] - \Pr [D(X_1 \ldots X_{i-1}Y_i \ldots Y_k) = 1] > \epsilon \tag{4.3}
\]

Now, \( X_1 \ldots X_{i-1} \) and \( Y_{i+1} \ldots Y_k \) will be fixed to \( \bar{x}_1 \ldots \bar{x}_{i-1} \) and \( \bar{y}_{i+1} \ldots \bar{y}_k \) respectively. Indistinguishability in the non-uniform setting refers to indistinguishability against (non-uniform) circuits. In this context, the (independent) values of \( \bar{x}_1, \ldots, \bar{x}_{i-1} \) and \( \bar{y}_{i+1}, \ldots, \bar{y}_k \) can be given as non-uniform advice to the distinguishing circuit \( D' \). Namely, this circuit besides its input has these values hard wired into it. \( D' \) for each input (of length \( n \)) uses \( D \) and the advice values for \( \bar{x}_1, \ldots, \bar{x}_{i-1}, \bar{y}_{i+1}, \ldots, \bar{y}_k \) to output one or zero. Now, as shown below, if \( D \) has high enough probability of distinguishing between two neighboring hybrids, \( D' \) can distinguish between two single samples with high probability as well. After fixing \( X_1 \ldots X_{i-1} \) and \( Y_{i+1} \ldots Y_k \) appropriately to \( \bar{x}_1 \ldots \bar{x}_{i-1} \) and \( \bar{y}_{i+1} \ldots \bar{y}_k \), a new distinguisher \( D' \) which distinguishes between a single sample of \( X \) and \( Y \) (namely, it

\*\( \text{(Otherwise, } \sum_{i=1}^{k} \Pr [D(H_i) = 1] - \Pr [D(H_{i-1}) = 1] \leq \frac{\epsilon k}{k} = \epsilon k \)\)
distinguishes between $X_i$ and $Y_i$) can be constructed as follows:

$$D'(x) = D(\bar{x}_1 \ldots \bar{x}_{i-1} x \bar{y}_{i+1} \ldots \bar{y}_k)$$

Given $X_i$ and $Y_i$, $D'$ distinguishes $X_i$ from $Y_i$ by computing $D'(X_i)$ and $D'(Y_i)$. Since

$$\text{Pr}[D(X_1 \ldots X_{i-1} X_i \ldots Y_k) = 1] - \text{Pr}[D(X_1 \ldots X_{i-1} Y_i \ldots Y_k) = 1] = \text{Pr}[D'(X_i) = 1] - \text{Pr}[D'(Y_i)] > \epsilon$$

$D'$ (which is still of polynomial size*) distinguishes between $X$ and $Y$ with probability greater than $\epsilon$ which is a contradiction to the indistinguishability assumption of the two distributions $X$ and $Y$.

**Remark 35** In the uniform setting, indistinguishability will be preserved under multiple samples only if we are able to efficiently (in polynomial time) sample the distributions. If polynomial time sampling of the distributions is possible, then the argument goes through with one modification. Namely, that the the new distinguisher $D'$ (a polynomial time algorithm) first efficiently samples $X_1 \ldots X_{i-1}, Y_{i+1} \ldots Y_k$ and then uses the polynomial time distinguisher $D$. Similar to the non-uniform scenario, if $D$ has high enough probability of distinguishing between two neighboring hybrids, it can be shown that $D'$ can also distinguish between two single samples.

### 4.2 Preservation of indistinguishability in the derandomization setting

In the derandomization setting, unlike the cryptography setting, giving the distinguisher more information does not jeopardize indistinguishability. For instance, if the first distribution $X$ is produced by a pseudorandom generator $G$ (e.g., the NW PRG) on a truly random seed $\sigma$ and the second distribution $Y$ is $U_n$, then giving out $\sigma$ still preserves indistinguishability because the distinguisher does not have enough computational resources to compute $G(\sigma)$. More precisely, in the derandomization setting, the generator is allowed to run in

---

*$|D'| = |D| + O(k)$. Since $D$ is a polynomial-size distinguisher and $k$ is a polynomial as well, $D'$ will be a polynomial-size distinguisher as well.
polynomial time in its output length whereas the distinguisher is strictly less powerful and
runs in time $n$ (where $n$ is the output length of the generator), and even if the adversary
completely receives $\sigma$, it’s not powerful enough to compute $G(\sigma)$. Hence, it can not distin-
guish between $G(\sigma)$ and the uniform distribution.

We are given two $\epsilon$-indistinguishable random variables $X$ and $Y$ which can be computed
given two seeds $\sigma$ and $\tau$ respectively, and furthermore $X$ and $Y$ can be thought of as
$(\sigma, G_\sigma(\sigma))$ and $(\tau, U_n)$. The goal is to show that if $(\sigma, X)$ and $(\tau, Y)$ are indistinguish-
able, then indistinguishability is preserved for multiple samples $(\sigma_1, X_1)$ ... $(\sigma_k, X_k)$ and
$(\tau_1, Y_1)$ ... $(\tau_k, Y_k)$ as well. Similar to the cryptography setting, we use hybrids to as-
sist us in deriving a contradiction. We assume that there is a distinguisher between
$(\sigma_1, X_1)$ ... $(\sigma_k, X_k)$ and $(\tau_1, Y_1)$ ... $(\tau_k, Y_k)$ with probability greater than $\epsilon k$
and derive a contradiction to $\epsilon$-indistinguishability of $X$ and $Y$. Denote the generating functions
of the two distributions $X$ and $Y$ by $f_x$ and $f_y$ such that $f_x(\sigma) = X$ and $f_y(\tau) = Y$.
Non-boolean functions $f_x$ and $f_y$ need to be hard to compute for indistinguishability of $X$ and
$Y$ to hold. Intuitively, one can think of $(\sigma, X)$ as the output of the NW PRG given seed
$\sigma$, and $(\tau, Y)$ as the uniform distribution. In this scenario, $|\tau|$ is super-polynomially larger
than $|\sigma|$ and $f_y(\tau) = U_n$.

Indistinguishability of $X$ and $Y$ implies that function $f_x$ and $f_y$ used in the construction
of $X$ and $Y$ should be very hard to be computed by the adversary.

- **Uniform setting**
  - Completely independent samples:
    As discussed in the cryptography setting, preservation of indistinguishability in
    the uniform setting, heavily depends on the efficiency of sampling; in other words,
    functions $f_x$, $f_y$ are efficiently computable. But, if that was the case, then the
    adversary could use $f_x$ to compute $X$: which is not possible. So, indistinguisha-
    bility is not preserved for multiple independent samples under the uniform setting.

- **Non-Uniform setting**
  In fact, in the derandomization setting with small dependence, we need to give out
  the seed to prove that indistinguishability is preserved under multiple samples. Here,
  in the hybrid argument $X_1$ ... $X_{i-1}$ and $Y_{i+1}$ ... $Y_k$ can not be fixed due to their
small dependence on $X_i$ and $Y_i$, and they become functions of a small string coming from $X_i$ and $Y_i$ respectively, and so we give out the portion of $\sigma_j$ that is in common with $\sigma_i$: $\sigma_j' = \sigma_j|_{X_i \cap X_j}$.

**Theorem 36** If $n$-bit random variables $(\sigma, X = f_x(\sigma))$ and $(\tau, Y = f_y(\tau))$ are $\epsilon$-indistinguishable for circuits of size $s$, then $(\sigma_1, X_1) \ldots (\sigma_k, X_k)$ and $(\tau_1, Y_1) \ldots (\tau_k, Y_k)$ are also $\kappa$-indistinguishable for circuits of size $s'$ where $s'$ depending on the amount of dependence between the samples is defined as:

- $s' = s - O(kn)$ if the samples are completely independent, or
- $s' = s - \frac{2^k}{\kappa}$ if seeds have at most $\kappa$ bits in common.

**Proof Sketch:** (Assume that there is a distinguisher $D$ of size $s'$ which distinguishes $(\sigma_1, X_1), \ldots, (\sigma_k, X_k)$ from $(\tau_1, X_1), \ldots, (\tau_k, Y_k)$ with probability greater than $\epsilon k$:

$$|\Pr[D((\sigma_1, X_1), \ldots, (\sigma_k, X_k)) = 1] - \Pr[D((\tau_1, X_1), \ldots, (\tau_k, Y_k)) = 1]| > \epsilon k$$

Again, without loss of generality, the absolute values can be removed:

$$\Pr[D((\sigma_1, X_1), \ldots, (\sigma_k, X_k)) = 1] - \Pr[D((\tau_1, X_1), \ldots, (\tau_k, Y_k)) = 1] > \epsilon k$$

Build hybrids $H_1, \ldots, H_k$ where

$$H_i = (\sigma_1, X_1) \ldots (\sigma_i, X_i) (\tau_{i+1}, Y_i+1) \ldots (\tau_k, Y_k).$$

So, $H_0 = (\tau_1, Y_1) \ldots (\tau_k, Y_k)$ and $H_k = (\sigma_1, X_1) \ldots (\sigma_k, X_k)$.

Success probability of distinguisher $D$ (stated above) can be rewritten in terms of the two extreme hybrids $H_0$ and $H_k$ as follows:

$$\Pr[D(H_k) = 1] - \Pr[D(H_0) = 1] > \epsilon k$$

$$\Sigma_{i=1}^k(\Pr[D(H_i) = 1] - \Pr[D(H_{i-1}) = 1]) > \epsilon k$$

$$\exists i, \Pr[D(H_i) = 1] - \Pr[D(H_{i-1}) = 1] > \frac{\epsilon k}{k} = \epsilon$$

We'd like to build a new distinguisher that, using $D$, distinguishes $(\sigma, X)$ from $(\tau, Y)$ and hence derives a contradiction (since $X$ and $Y$ by assumption are computationally indistinguishable). Expand the hybrid notation to get:
\[
\Pr[D((\sigma_1, X_1) \ldots (\sigma_{i-1}, X_{i-1})(\sigma_i, X_i) \ldots (\tau_k, Y_k)) = 1] - \\
\Pr[D((\sigma_1, X_1) \ldots (\sigma_{i-1}, X_{i-1})(\tau_i, Y_i) \ldots (\tau_k, Y_k)) = 1] > \epsilon
\] (4.4)

Now, the amount of dependence between the multiple samples determines how \(X_1 \ldots X_{i-1}\) and \(Y_{i+1} \ldots Y_k\) will be fixed. This fixing enables another distinguisher \(D'\) (of size \(s\)) to derive a contradiction by distinguishing \(X_i\) from \(Y_i\) with probability more than \(\epsilon\).

- **Completely independent samples:**
  
  Now, since \(X_j\)'s (and similarly \(Y_j\)) are completely independent from each other, using an averaging argument one can fix \((\sigma_1, X_1) \ldots (\sigma_{i-1}, X_{i-1})\) and \((\tau_{i+1}, Y_{i+1}) \ldots (\tau_k, Y_k)\) to \(\tilde{a}_1 \ldots \tilde{a}_{i-1} \tilde{a}_{i+1} \ldots \tilde{a}_k\) and still preserve inequality (4.4). In this context, \(\tilde{a}_1 \ldots \tilde{a}_{i-1} \tilde{a}_{i+1} \ldots \tilde{a}_k\) can be given as non-uniform advice to the distinguishing circuit \(D'\). Namely, this circuit besides its input has these values hard wired into it. \(D'\) for each input (of length \(n\)) uses \(D\) and the advice values for \((\sigma_1, X_1), \ldots, (\sigma_{i-1}, X_{i-1}), (\tau_{i+1}, Y_{i+1}), \ldots, (\tau_k, Y_k)\) to output 1 or 0. Now, if \(D\) has high enough probability of distinguishing between two neighboring hybrids, \(D'\) can also distinguish between two single samples \(X_i\) and \(Y_i\), where \(D'\) is of size \(s' = |D| + O(kn) = s - O(kn) + O(kn) = s\).

- **Samples with small dependence:**
  
  In this setting, the \(k\) samples are not entirely independent rather they have small dependence among them. In other words, any \(\sigma_j\) and \(\sigma_i\) (and similarly \(\tau_j\) and \(\tau_i\) for \(i \neq j\)) have small dependence (bits in common). Now, since \(X_j\)'s (and similarly \(Y_j\)'s) are *not* completely independent from each other (due the dependence between \(\sigma_k\)'s and \(\tau_k\)'s respectively), we can not completely fix \(X_1 \ldots X_{i-1}\) and \(Y_{i+1} \ldots Y_k\) by fixing their seeds. We focus on the \(NW\)-type dependence. Namely, \(NW(\sigma) = (\sigma_1, \ldots, \sigma_k)\) and \(NW(\tau) = (\tau_1, \ldots, \tau_k)\) where \(\sigma\) and \(\tau\) are chosen uniformly random and the intersection between \(\sigma_i\)'s is bounded by \(\hat{x}\), and similarly by \(\hat{y}\) for \(\tau_i\)'s. Despite lack of *complete* independence, because there's small dependence between the multiple samples, the bits \(\sigma_j\)'s (for any \(j \neq i\)) that is not in common with \(\sigma_i\) can be fixed. In other words, after fixing the bits that are not dependent on \(\sigma_i\), each \(\sigma_j\) becomes a function of \(\hat{x}\) bits and
simply, each \( r_j \) becomes a function of \( \hat{y} \) bits. As it will be shown later (Lemma 38), this will be used to construct a distinguisher \( D' \) of size \( \frac{2^k}{2} + |D| = \frac{2^k}{2} + s. \)

**Definition 37 (Indistinguishability-Preserving) [HV04]** A generator

\[ \{0,1\}^t \to (\{0,1\}^n)^k \]

is said to be indistinguishability-preserving for size \( t \) if for all (possibly probabilistic) functions \( f_1, \ldots, f_k, g_1, \ldots, g_k \) the following holds:

If for every \( i, 1 \leq i \leq k \) the distributions \( x_{f_i}(x) \) and \( y_{g_i}(y) \) are \( \epsilon \)-indistinguishable for size \( s \) where \( x \in U_n \) and \( y \in U_n \), then \( \sigma f_1(X_1) \ldots f_k(X_k) \) and \( \sigma g_1(X_1) \ldots g_k(X_k) \) are \( k \epsilon \)-indistinguishable for size \( s - t \) where \( \sigma \in U_l \) and \( X_1, \ldots, X_k \) is the output of \( G(\sigma) \).

Based on Definition 37, an example of an indistinguishability-preserving generator is the NW pseudorandom generator: \( NW(x) = (x_{S_1}, \ldots, x_{S_k}) \) where \( S_1, \ldots, S_k \) form a combinatorial design. Healy, Vadhan, and Viola [HV04] show that the NW generator (even before the application of the hard function) is indistinguishability-preserving for size \( k^2 \mu + 31 \ln 31. \)

**Lemma 38 [HV04]** There is a constant \( c \) such that for every \( n \geq 2 \) and \( k = k(n) \), there is a generator \( NW : \{0,1\}^l \to (\{0,1\}^n)^k \) defined by \( NW(x) = (x_1 \ldots x_k) \), where \( S_1, \ldots, S_k \) are \((\log(k), n)\) designs with seed-length \( l = O(n) \) and each \( x_i = x|_{S_i} \), that is indistinguishability-preserving for size \( k^2 \).

**Proof Sketch:** To prove that \( NW \) is indistinguishability-preserving for size \( k^2 \) (based on Definition 37), one has to show that for all functions \( f_1 \ldots f_k, g_1 \ldots g_k \) that (given truly random seeds) are \( \epsilon \)-indistinguishable for size \( s \), \( \sigma f_1(x_1) \ldots f_k(x_k) \) is \( k \epsilon \)-indistinguishable from \( \sigma g_1(x_1) \ldots g_k(x_k) \) for size \( s - k^2 \).

By preservation of indistinguishability in the derandomization setting (described in Section 4.2) with respect to non-uniform distinguishers, it follows that if two ensembles \( E_F = f_i_{i=1}^k \) and \( E_G = g_i_{i=1}^k \) are element-by-element \( \epsilon \)-indistinguishable, then \( k \) samples from them will be \( k \epsilon \)-indistinguishable with respect to slightly smaller circuits.

Let's assume that there is a circuit of size \( s - k^2 \) that distinguishes \( \sigma f_1(x_1) \ldots f_k(x_k) \) from \( \sigma g_1(x_1) \ldots g_k(x_k) \) with probability greater than \( \epsilon k \). In other words, without loss of generality the following holds:

\[ \Pr[C(\sigma f_1(x_1) \ldots f_k(x_k))] = 1 - \Pr[C(\sigma g_1(x_1) \ldots g_k(x_k))] = 1 > \epsilon k \]

* A function of input size \( n \) can be computed by a circuit of size at most \( \frac{2^k}{n} \). [Sha49]
Using the same reasoning as in the non-uniform setting of Section 4.2, one can build hybrids and use non-uniformity to prove that the previous inequality leads to another distinguishing circuit \(C'\) of size \(|C| + k^2 = s\) that distinguishes \(f_i\) from \(g_i\); an apparent contradiction.

**Proof:** We will prove that for \((x_1, \ldots, x_k) = NW(x)\) (where \(x_1, \ldots, x_k\) are \((n, \log(k))\) combinatorial designs) and for all circuits of size \(s - k^2\), \((x_1, f(x_1)) \ldots (x_k, f(x_k))\) is \(\epsilon k\)-indistinguishable from \((x_1, g(x_1)) \ldots (x_k, g(x_k))\). Assume there is a circuit \(C'\) of size \(s - k^2m\) such that

\[
\Pr[C'(x_1, f(x_1)) \ldots (x_k, f(x_k))) = 1] - \Pr[C'(x_1, g(x_1)) \ldots (x_k, g(x_k))) = 1] \geq \epsilon k
\]

Build hybrids \(H_0, \ldots, H_k:\)

\[
H_j = (x_1, f(x_1)) \ldots (x_j, f(x_j))(x_{j+1}, g(x_{j+1})) \ldots (x_k, g(x_k)).
\]

By definition, \(H_0 = (x_1, g(x_1)) \ldots (x_k, g(x_k))\) and \(H_k = (x_1, f(x_1)) \ldots (x_k, f(x_k))\).

\[
\Pr[C'(x_1, f(x_1)) \ldots (x_k, f(x_k))) = 1] - \Pr[C'(x_1, g(x_1)) \ldots (x_k, g(x_k))) = 1] \geq \epsilon k
\]

By the hybrid argument, there exists \(i\) such that:

\[
\Pr[C'(H_{i-1}) = 1] - \Pr[C'(H_i) = 1] \geq \frac{\epsilon k}{k} = \epsilon
\]

For this \(i\),

\[
\Pr[C'(x_1, f(x_1)) \ldots (x_{i-1}, f(x_{i-1}))(x_i, g(x_i)) \ldots (x_k, g(x_k))) = 1]
- \Pr[C'(x_1, f(x_1)) \ldots (x_{i-1}, f(x_{i-1}))(x_i, f(x_i)) \ldots (x_k, g(x_k))) = 1] \geq \epsilon
\]

Similar to arguments in preservation of indistinguishability in Section 4.2 and by an averaging argument, we try to fix \((x_1, f(x_1)) \ldots (x_{i-2}, f(x_{i-2}))\) and \((x_{i+1}, g(x_{i+1})) \ldots (x_k, g(x_k))\). Because of the small pairwise dependence between \(x_1, \ldots, x_k\), we can not completely fix them, we can however, using an averaging argument, fix the bits of \(x_1 \ldots x_{i-1}x_{i+1} \ldots x_k\) that are not in common with \(x_i\). Since the common bits between any two \(x_i\) and \(x_j\) is smaller than \(\log k\), there (potentially) remain \(n - \log k\) variable bits. So, now each \((x_j, f(x_j))\) turns into \((\hat{x}_j, f(\hat{x}_j))\) where \(\hat{f}\) receives a smaller input, has as non-uniform advice a truth table of size \(2^{\log(k)}\) for the fixed bits, and outputs
CHAPTER 4. PRESERVATION OF INDISTINGUISHABILITY

$f(x_j)$. Since $f(x_j)$ is a boolean function, and there are potentially $k$ such calculations, the size of the distinguisher is at most $2^{\log k} k = k^2$. So, distinguisher $C''$ (of size $k^2$) can distinguish $(x_i, f(x_i))$ from $(x_i, g(x_i))$ with probability greater than $\epsilon$ which contradicts our indistinguishability assumption.

The previous argument requires that the circuits computing $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k$ are very small. The size of such circuits is upper bounded by $\frac{2^{\nu}}{\nu}$ [Sha49] (where $\nu$ is the input length of the boolean function). To ensure that the size of the circuits is small (e.g., polynomial), $\nu$ will have to be very small (respectively logarithmic). We generalize this result of [HV04] in the following theorem.

**Theorem 39** For every $k = k(n)$, denote $NW(s) : \{0,1\}^l \rightarrow (\{0,1\}^n)^k$ by $NW(\sigma) = (x_1 \ldots x_k)$, where $S_1 \ldots S_k$ are $(\log k, n)$ designs with seed-length $l = O(n)$ and each $x_i = x|S_i$. Let $f_x$ and $g_x$ be random variables from $\{0,1\}^n$ to $\{0,1\}^m$.

If $\sigma' f_x(\sigma')$ and $\sigma' f_y(\sigma')$ are $\epsilon$-indistinguishable for circuits of size $s$, then $\sigma f_x(x_1) \ldots f_x(x_k)$ are $c k$-indistinguishable from $\sigma f_y(x_1) \ldots f_y(x_k)$ for circuits of size $s - k^2 m$.

**Proof:** The argument is analogous to that of Lemma 38, and the only difference would be the size of the advice given. By an averaging argument, we try to fix $(x_1, f(x_1)) \ldots (x_{i-2}, f(x_{i-2}))$ and $(x_{i+1}, g(x_{i+1})) \ldots (x_k, g(x_k))$. However, because of the small pairwise dependence between $x_1, \ldots, x_k$, we can not completely fix them, we can however, using an averaging argument, fix the bits of $x_1 \ldots x_{i-1} \ldots x_{i+1} \ldots x_k$ that are not in common with $x_i$. Since the common bits between any two $x_i$ and $x_j$ is smaller than $\log k$, there remain $n - \log k$ variable bits. So, now each $(x_j, f(x_j))$ turns into $(\tilde{x}_j, \tilde{f}(\tilde{x}_j))$ where $\tilde{f}$ is a function of $\log k$ bits. There are potentially $k$ such calculations of $\tilde{f}$. As a result, the size of the distinguisher is at most $2^{\log k} mk = k^2 m$, and as long as $k$ and $m$ are small functions, the produced distinguisher $C''$ of size $k^2 m$ is small enough and given $(x_i, f(x_i))$ and $(x_i, g(x_i))$, it can distinguish them with probability greater than $\epsilon$ which contradicts our indistinguishability assumption.

We can not compute $f_j$ but we can represent it with a look up table of size $2^a$ where $a$ denotes the size of the maximum bits in common between $x_i$'s which can be included in the advice to the non-uniform algorithm. ■
Chapter 5

Correctness of our construction

In this section we prove our main result, namely that the output of the construction outlined in Section 3 produces a computationally pseudorandom distribution. Recall that $N$ denotes $2^n$.

**Theorem 40** Let $P$ be a $O(n)$-bit $\delta$-hard predicate with respect to circuits of size $2^{O(n)}$, $z$ be a uniformly chosen string from $U_{O(n)}$, and denote the output of $NW^{long}(z)$ by $(z_1, \ldots, z_N)$ where $|z_i| = O(n)$ for all $i$. Then, generator $G(z) : \{0,1\}^{O(n)} \rightarrow \{0,1\}^{\Theta(nN)}$ defined as $G(z) = EXT(P(NW^{short} \oplus EW)(z_1)) \ldots EXT(P(NW^{short} \oplus EW)(z_N))$ is a $\frac{1}{\Omega(d)}$-PRG for size $O(N^{O(1)})$.

The theorem relies on observations and proof techniques of Sudan, Trevisan and Vadhan in [STV01]. Our proof structure is inspired by, but slightly different from their proof. In particular, they prove that the output of their generator (see Figure 1.5) is computationally indistinguishable from an intermediate distribution $D$ which in turn is statistically close to a distribution $D'$ of high min-entropy. They combine the properties of the NW generator along with a pairwise independent generator to prove indistinguishability. We modularize the proof of Theorem 40 and use two intermediate distributions $E_y$ and $E_x$ which can be considered as restrictions of the intermediate distributions $D$ and $D'$ used in [STV01].

Denote the density functions of distributions $D$ and $D'$ by $f_D(x, y)$ and $f_{D'}(x, y)$ respectively. $E_y$ is the distribution obtained by $f_D(x, \hat{y})$ and, similarly, $E_x$ is the distribution defined by $f_{D'}(\hat{x}, y)$ where $\hat{x}$ and $\hat{y}$ are fixed strings, and $x$ and $y$ vary over $\{0,1\}^n$. Our proof can be outlined in the following several steps:

1. We show that, given truly random seeds, our second layer of construction is a collection
of weak sources (denoted by $T_i$'s in Figure 3.2) with high min-entropy. In particular, we prove that its output is computationally indistinguishable from a distribution that is a convex combination of oblivious bit fixing sources with high min-entropy.

2. We then prove that using $NW^{long}$ as a generator of seeds for the second layer (instead of $N$ truly random seeds) preserves computational indistinguishability between $\text{EXT}(T_1) \ldots \text{EXT}(T_N)$ (recall Definition 33) and $\hat{U}_1 \ldots \hat{U}_N$ where $\hat{U}_i \in U_{\theta(n)}$.

For clarity, we use two distinct notations for the sources of our construction's second layer based on the quality of their given seeds. More precisely, if the sources in the second layer are given the seeds coming from $NW^{long}$, then we denote them as $T_i$'s (as stated on Figure 3.2) whereas if their given seeds are truly random seeds taken from $U_{O(n)}$, we denote each such source by $T'_i$.

**Definition 41** Define a slight modification of the sources $T_i$ (recall Definition 33) in the second layer of our construction (Figure 3.2) as $T'_i = (P(x_i^1 \oplus y_i^1) \ldots P(x_i^n \oplus y_i^n))$ where

- $x_i^j \in U_{O(n)}$,
- $x_i^j y_i^j = z_i^j$,
- $\text{EW}(y'_i) = (y_i^1, \ldots, y_i^n)$,
- $NW^{short}(x_i) = (x_i^1, \ldots, x_i^n)$.

In what follows, we show that each $T'_i$ is computationally indistinguishable from (a convex combination of) oblivious bit fixing sources. This proof of indistinguishability takes advantage of Impagliazzo's hard-core lemma (see Lemma 17) which states that a boolean function that circuits of a certain size can not correctly compute on a constant fraction of inputs has a subset of instances which circuits of slightly smaller size can not compute any better than a random guess.

For each source $T'_i$, we prove the following theorem of indistinguishability.

**Theorem 42** Let $NW(x^i) = (x_i^1, \ldots, x_i^n)$ and $\text{EW}(y^i) = (y_i^1, \ldots, y_i^n)$ where $x_i^j \in U_{O(n)}$ and $y_i^j \in U_{O(n)}$. Then, given a predicate $P : \{0,1\}^{O(n)} \rightarrow \{0,1\}$ that is $\delta$-hard for size $s$, distribution $T'_i(x^i, y^i) = P(x_i^1 \oplus y_i^1) \ldots P(x_i^n \oplus y_i^n)$ is $\frac{1}{\delta n^2} \cdot \frac{1}{2^s}$-indistinguishable from a convex combination of $(n,\frac{\delta n}{2})$-oblivious bit fixing sources for size $s' = O\left(\frac{1}{n^2 \cdot 2^n}\right)s - O(n^2)$.
CHAPTER 5. CORRECTNESS OF OUR CONSTRUCTION

Proof: Let $H$ denote the hard-core set of $\delta$-hard predicate $P$ as guaranteed by Impagliazzo’s Hard-core lemma (Lemma 17). Here, we prove that each source $\Gamma'_i$ is computationally indistinguishable from distribution $D$ defined as:

Definition 43 (Distribution $D$)

- Given: $(x, y)$ where $|x| = |y| = n$. Denote the $n$ bits of $x$ (and analogously $y$) by $x_1 \ldots x_n$ (and respectively $y_1 \ldots y_n$). In our context, $x$ is the output of $\text{NW}_{\text{short}}$ given a truly random seed, and $y$ is generated by $\text{EW}$ using a truly random seed.

- Output: $b_1, \ldots, b_n$ where
  
  $$b_i = \begin{cases} 
  U_1 & x_i \oplus y_i \in H \\
  P(x_i \oplus y_i) & x_i \oplus y_i \notin H 
  \end{cases}$$

For succinctness and more readability, we organize our proof that each $\Gamma'_i$ source is indistinguishable from a convex combination of oblivious bit-fixing sources into two modules. This modularized proof further illuminates roles of each generator $\text{NW}_{\text{short}}$ and $\text{EW}$, and emphasizes the necessity of using each generator. Throughout this proof, let $\text{NW}(x) = (x_1 \ldots x_n)$ and $\text{EW}(y) = (y_1 \ldots y_n)$ where $x \in U_{O(n)}$ and $y \in U_{O(n)}$.

Since $P$ is a $\delta$-hard boolean function for circuits of size $s$, Lemma 17 guarantees the existence of a $\frac{1}{n^{\delta n}}$-hard-core set $H$. More specifically, there exists $H \subseteq \{0, 1\}^{O(n)}$ such that $|H| = \delta 2^{O(n)}$ and for any circuit $C$ of size at most $s' = O(\frac{1}{n^{\delta n}} s)$,

$$\Pr_{x \in U_{H}}[C(x) = P(x)] \leq \frac{1}{2} + \frac{1}{n^{2\delta n}}.$$  \hspace{1cm} (5.1)

1. First Module: Computational indistinguishability ensured by $\text{NW}_{\text{short}}$

Fix $O(n)$-bit string $\tilde{y}$. Let $\text{EW}(\tilde{y}) = (\tilde{y}_1, \ldots, \tilde{y}_n)$ and define $H^\dagger_{\tilde{y}} = H \oplus \tilde{y}_i$.

Definition 44 Distribution $E_{\tilde{y}}$:

$$b_i = \begin{cases} 
  U_1 & x_i \in H^\dagger_{\tilde{y}} \\
  P^\dagger_{\tilde{y}}(x_i) = P(x_i \oplus \tilde{y}_i) & x_i \notin H^\dagger_{\tilde{y}} 
  \end{cases}$$

Output $b_1, \ldots, b_n$.

Distribution $E_{\tilde{y}}$ and $D$ behave similarly in the sense that $(x_i \oplus \tilde{y}_i) \in H$ if and only if $x_i \in H^\dagger_{\tilde{y}}$. We prove that distribution $E_{\tilde{y}}$ as defined in Definition 44 is computationally indistinguishable from $D$.
indistinguishable from the output of \( T'_i(x^i, \hat{y}) \) with respect to circuits of size at most \( O\left(\frac{s}{n^{2\gamma}}\right) - O(n^2) \). This also implies that each \( T' \) is computationally indistinguishable from \( D \) since our argument holds for any \( \hat{y} \in \{0, 1\}^{O(n)} \). We argue this indistinguishability proof in terms of distribution \( E_\hat{y} \) to emphasize that this theorem holds due to the special properties of the designs and that generator \( EW \) doesn't play a vital role in this step.

**Claim 45** No circuit of size \( O\left(\frac{1}{n^{2\gamma}}\right)s - O(n^2) \) can distinguish between the output of \( T'_i(x^i, \hat{y}) = (P(x^i_1 \oplus \hat{y}_1) \ldots P(x^i_n \oplus \hat{y}_n)) \) and \( E_\hat{y} \) with advantage more than \( \frac{1}{n^{2\gamma}} \).

**Proof:** [Claim 45]

If \( x^i_j \) is sampled uniformly from complement of \( H^i_j \), then distributions \( T'_i(x^i_j, \hat{y}) \) and \( E_\hat{y} \) are indistinguishable since, on a given \( x^i_j \), they both produce \( P(x^i_j \oplus \hat{y}_j) \). If \( x^i_j \) is sampled uniformly from \( H^i_j \), then \( P(x^i_j \oplus \hat{y}_j) \) generates a distribution that is indistinguishable from a random bit. Precisely, distribution \( \{P(x^i_j \oplus \hat{y}_j)\}_{x^i_j \in H^i_j} \) is \( \left(\frac{1}{n^{2\gamma}} - \frac{1}{2}\right) \)-indistinguishable from \( U_1 \) (by inequality (5.1)) with respect to circuits of size \( s' = \frac{1}{n^{2\gamma}}s \). Thus, distribution \( \{T'_i(x^i_j, \hat{y}_j)\}_{x^i_j \in H^i_j} \) will be \( \left(\frac{1}{n^{2\gamma}} - \frac{1}{2}\right) \)-indistinguishable from \( \{E_\hat{y}(x^i_j)\}_{x^i_j \in H^i_j} \) with respect to circuits of size \( s' \). Now, by Lemma 38, it follows that \( \{P(x^i_1 \oplus \hat{y}_1) \ldots P(x^i_n \oplus \hat{y}_n)\} \) (where \( x^i_j \)s are sampled uniformly from \( U_{O(n)} \)) will be \( \left(\frac{1}{n^{2\gamma}} - \frac{1}{2}\right) \)-indistinguishable for circuits of size \( s' - O(n^2) \).

We then prove that distribution \( D \) described above is statistically close to distribution \( D' \) defined (see Definition 46) below. Moreover, we show that \( D' \) is a convex combination of oblivious bit fixing sources.

**Definition 46 (Distribution \( D' \))** Distribution \( D' \):

- **Given:** \((x, y)\) where \(|x| = |y| = n\). Denote the \( n \) bits of \( x \) (and analogously \( y \)) by \( x_1 \ldots x_n \) (and respectively \( y_1 \ldots y_n \)). In the context of our construction and proof, \( x \) is the output of \( NW_{\text{short}} \) given a truly random seed, and \( y \) is generated by \( EW \) using a truly random seed.

- **Output:**
  - **Case 1:** If the number of \((x_1 \oplus y_1, \ldots, x_n \oplus y_n)\) that fall into \( H \) are smaller than \( \frac{\delta_n}{2} \), then Output \( U_n \). This case, as proven later, will occur only with exponentially small probability.
CHAPTER 5. CORRECTNESS OF OUR CONSTRUCTION

- **Case 2:** If the number of \((x_1 \oplus y_1, \ldots, x_n \oplus y_n)\) that are in \(H\) is at least \(\frac{\delta n}{2}\), then output according to distribution \(D\) defined above.

2. **Second module:** High min-entropy ensured by \(EW\)

Fix \(\hat{x} = (x_1, x_2, \ldots, x_n) = NW_{x \in U_{\Omega(n)}}(x)\), and define \(H'_{2} = H \oplus x_i\).

**Definition 47** Define Distribution \(E_{2}\) as

\[
\begin{cases}
U_n \\
b_1 \ldots b_n \\
where \ b_n = \begin{cases}
U_1 \\
y_i \in H^i_{\hat{x}} \\
P_i^\hat{x} = P(x_i \oplus y_i) \\
y_i \notin H^i_{\hat{x}}
\end{cases}
\end{cases}
\]

Distribution \(E_2\) and \(D'\) behave similarly in the following sense: If \((\hat{x}_i \oplus y_i) \in H\) then \(\hat{x}_i \oplus (\hat{x}_i \oplus y_i) \in \hat{x}_i \oplus H\) and \(y_i \in H^i_{\hat{x}}\), and vice versa. In other words, \(x_i \oplus y_i \in H\) if and only if \(y_i \in H^i_{\hat{x}}\), and \(|H| = |H^i_{\hat{x}}|\) which implies that \(D'\) and \(E_2\) both have the same amount of min-entropy. \(E_2\) is an oblivious bit-fixing source because for the indices \(j\) that \(y_j \in H_{\hat{x}}, y_j's\) are computationally indistinguishable from the uniform distribution, and for any \(j\) such that \(y_j \notin H_{\hat{x}}, y_j\) is fixed to \(P(y_j \oplus \hat{x}_j^i)\).

Since \(D'\) is a convex combination of \(E_2\) sources (where each \(\hat{x}\) is chosen uniformly at random) and for each \(\hat{x}, E_2\) is an oblivious bit-fixing source, it follows that \(D'\) is also a convex combination of oblivious bit fixing sources.

We argue the proof in terms of distribution \(E_2\) to emphasize that this proof goes through because the \(EW\) generator ensures that several \(y_i\)'s fall into the hard-core set, and that the designs do not play a crucial role in this part of the proof.

**Claim 48** Distribution \(D'\) is a convex combination of oblivious bit fixing sources of min-entropy \(\frac{\delta n}{2}\).

Recall that \(D'\) is a convex combination of \(X_1, \ldots, X_N\) if there is a distribution \(I\) on \(\{1, \ldots, N\}\) such that \(D'\) is realized by choosing \(i\) according to \(I\) and taking a sample \(\psi\) from \(X_i\) and outputting \(\psi\). Let \(I\) be the uniform distribution over all strings possible for \(\hat{x}\). Namely, \(I = U_{\Omega(n^2)}\), and \(D'\) can be realized by taking a convex combination of possible \(E_2\)s using \(I\).
Distribution \( E_z \) (as defined in Definition 47) trivially has \( \frac{4n}{2} \) min-entropy. In the first case (which occurs with exponentially small (in \( n \)) probability), the output distribution has \( \frac{4n}{2} \) min-entropy. In the second case, Theorem 31 states that \( EW \) has the property that almost certainly for \( \delta \geq \frac{1}{6} \), \( \Pr \left[ |\{i \mid y_i \in H_\delta\}| < \frac{4n}{2} \right] \leq 2^{-\frac{4n}{2}}\). This implies that \( (E_z \text{ and in turn}) \) \( D' \) is statistically (at most) \( 2^{-\frac{4n}{2}} \)-indistinguishable from \( D \) since they only differ in the case which happens with exponentially small probability. Thus, the advantage of any circuit in distinguishing \( D' \) from the output of \( \mathcal{T}_i \) is at most its (computational) advantage in distinguishing \( D \) from the output of \( \mathcal{T}_i \) plus the statistical difference between \( D \) and \( D' \). So, any circuit of size (at most) \( s \) has advantage \( O(\frac{1}{2^{|\mathcal{G}|}}) \) distinguishing \( \mathcal{T}_i \).

**Corollary 49** Given a \( \delta \)-hard predicate \( P \) for size \( s = 2^{O(n)} \), \( \mathcal{T}_i \) is \( O(\frac{1}{2^{|\mathcal{G}|}}) \)-indistinguishable from a convex combination of \( (n, \frac{4n}{2}) \)-oblivious bit fixing sources for size \( O(2^{|\mathcal{G}|}) - O(n^2) \) (where \( c < 1 \)).

This Corollary proves that the second layer of our construction is computationally indistinguishable from a convex combination of oblivious bit fixing sources if given uniformly random seeds. Next, we prove that indistinguishability will be preserved even if there is small dependence between the seeds.

**Theorem 50** Let explicit extractor \( EXT : \{0,1\}^n \rightarrow \{0,1\}^{(1-o(1))\delta n} \) be the oblivious bit-fixing extractor in Theorem 27. Then, \( G'_i(x) = EXT(\mathcal{T}_i(x)) \) (where \( x \in U_{O(n)} \)) is \( O(\frac{1}{2^{O(n)}}) \)-indistinguishable from \( U_{(1-o(1))\delta n} \) for circuits of size \( s' = O(2^n) - O(n^2) - n^{O(1)} \).

**Proof Sketch:** Similar to a proof in [STV01], suppose there is a distinguishing circuit \( C \) of size \( s' \) that distinguishes between \( G'_i \) and the uniform distribution. Using \( C \) build a circuit \( C' \) that has the extractor (which is of polynomial size) built in (i.e., \( C'(y) = C(EXT(y)) \)). Since \( C \) distinguishes between \( G' \) and the uniform distribution, by construction, \( C'(\mathcal{T}_i) = C(EXT(\mathcal{T}_i)) \) can be distinguished from \( C'(D') = C(EXT(D')) = C(U_{(1-o(1))\delta n}) \) where \( D' \) is the distribution of high min-entropy (defined in Definition 46) that is statistically close to \( \mathcal{T}_i \) (this is true because the extractor on a distribution of high min-entropy outputs a distribution that is statistically close to the uniform distribution). This implies that there is circuit \( C' \) that distinguishes \( D' \) from \( \mathcal{T}_i \) which is a contradiction by Corollary 49. \( \square \)
For simplicity, assume that these sources are oblivious bit fixing sources even though they are truly convex combinations of such sources. The proof goes through for the convex combination of oblivious bit fixing sources as well. This proof shows that using combinatorial designs in our construction’s first layer (see Figure 3.2) does not deteriorate the quality of the output of the generator. Since our ultimate goal of constructing such weak sources was to extract pseudorandom bits by applying extractors, we prove that \((\text{EXT}(P[(\text{NW}^{\text{short}} \oplus \text{EW})](z_1))] \ldots \text{EXT}(P[(\text{NW}^{\text{short}} \oplus \text{EW})](z_N))]\) is computationally indistinguishable from \(N\) copies of \(U_{O(n)}\). The significance of the new claim is that it demonstrates the power and stability of extractors. Namely, here we prove that extractors are powerful enough to extract randomness from a distribution that is computationally close to a weak source.

**Theorem 51** Let \(P\) be a \(\delta\)-hard predicate for size \(s\), \(\text{NW}^{\text{long}}(x)_{z \in U_{O(n)}} = (z_1, \ldots, z_N)\).

\((\text{EXT}(P[(\text{NW}^{\text{short}} \oplus \text{EW})](z_1))] \ldots \text{EXT}(P[(\text{NW}^{\text{short}} \oplus \text{EW})](z_N))]\) is \(O(\frac{\delta^2}{n^2 \log n})\)-indistinguishable for size \(O(\frac{\delta^2}{n^2 \log n})s - O(n^2) - N^2\) from \(N\) copies of \(U_{O(n)}\).

**Proof:** Denote computational indistinguishability with respect to circuits of size \(s\) by \(\cong_s\), and statistical closeness by \(\approx\). By Theorem 42, each \(P[(\text{NW}^{\text{short}} \oplus \text{EW})](x_i)]_{x_i \in U_{O(n)}}\) is computationally indistinguishable from distribution \(D\) for size \(\bar{s} = O(\frac{\delta^2}{n^2 \log n})s - O(n^2)\), and this distribution is in turn statistically close to a convex combination of oblivious bit fixing sources. In other words, for all \(i\), \(\text{EXT}(P[(\text{NW}^{\text{short}} \oplus \text{EW})](x_i)]_{x_i \in U_{O(n)}}) \cong_{\bar{s} - t} U_{(1-o(1))\delta n}\) (by Theorem 50) where \(\text{EXT}\) is the [GRS04] oblivious bit fixing extractor and \(t = n^{O(1)}\).

Let \(m\) denote \((1 - o(1))\delta n\).

By Definition 37 and Theorem 39, it follows that since since \(m < O(n) < N\) and

\[\sigma \in U_{O(n)}, \sigma \text{EXT}((\text{NW}^{\text{short}} \oplus \text{EW})(\sigma)) \cong_{\bar{s} - t} \sigma U_m,\]

Then, for \(x \in U_{O(n)}\), and \(x_1, \ldots, x_N \in \text{NW}^{\text{long}}(x)\),

\[x_1 \ldots x_N \text{EXT}(\text{NW}^{\text{short}} \oplus \text{EW})(x_1) \ldots \text{EXT}(\text{NW}^{\text{short}} \oplus \text{EW})(x_N) \cong_{\bar{s} - t - O(N^3)} x_1 \ldots x_N U_{\text{NO}(n)}.\]

Our main Theorem 40 follows as a corollary of the above theorem. Namely, the output of our generator is \(O(\frac{1}{\sqrt{\log n}}) N = O(\frac{1}{\sqrt{\log n}})\)-indistinguishable from \(N\) copies of \(U_m\) where \(m = (1 - o(1))O(\delta n)\). Hence, this generator is a \(\frac{1}{N^{O(1)}}\)-PRG for size \(O(N^{O(1)})\).
Chapter 6

Conclusion

Even though randomness is necessary in applications such as cryptography, tremendous effort has been initiated in complexity theory to demonstrate that it is not as critical in other applications such as algorithm design. Combinatorial constructions such as extractors and pseudorandom generators (though conditional on plausible complexity-theoretical assumptions) provide some evidence that randomness may be reduced and ultimately eliminated from randomized algorithms. This is a relief because in practice there are no truly random sources, and randomized algorithms have to rely on the "goodness" of ad-hoc random sources. Furthermore, even if randomness can not be completely excavated from algorithm designs, showing $\text{BPP} = \text{P}$ is a fundamental complexity question with fundamental implications in other fields of theoretical computer science.

Pseudorandom generator constructions of Nisan and Wigderson [NW94], followed by [IW97] and [STV01] provide great tools which take us a step closer to resolving this fundamental question. Using a constant-hard function, we construct a pseudorandom generator whose seed length is linear in the the input length of the hard function, a qualitative improvement on the parameters of [STV01]. Furthermore, this generator stretches its seed of length $O(n)$ to $O(n2^n)$ (computationally) pseudorandom bits. Our construction (Section 3) combines and reinterprets ideas from both constructions in [IW97] and [STV01]. We recast the output of Impagliazzo and Wigderson's generator (constructed for derandomizing Yao's XOR Lemma) as an oblivious bit-fixing source, a special weak source, and then using a very effective extractor [GRS04], we extract almost all of the randomness present in the output. Our construction, then, utilizes combinatorial designs to yield more pseudorandom bits.

We prove the correctness of our construction (Section 5) in stages. The output of each
source in our second layer of our construction (see Figure 3.2), when provided with truly random seeds, is suitable for the application of an oblivious bit-fixing extractor. We apply such an extractor ([GRS04]) to a distribution that is computationally indistinguishable from each of the constructed sources in the second layer and prove that the resulting distribution is indistinguishable from the uniform distribution. Finally, we use combinatorial designs to generate many such distributions and show that indistinguishability is preserved even after eliminating complete independence by using combinatorial designs.

Our second and third layers combined together form a distribution that is computationally indistinguishable from the uniform distribution, and by using the particular combinatorial designs of the first layer, we were able to argue that generating many such distributions without using complete independence yields a pseudorandom distribution. This raises the question of whether or not this technique would still yield a pseudorandom distribution when using an alternative to our second and third layers. Also, based on the interconnection between PRGs and extractors, it would be interesting to see if our PRG construction can be used to improve on existing extractor construction parameters. Finally, we used combinatorial designs in different layers of our construction. It would be interesting to analyze the properties of such designs in other applications as they have proven to be valuable in constructions of PRGs, weak sources, and in derandomization of direct products.
Appendix A

Auxiliary Proofs

A.1 Hardness vs. Indistinguishability

In this section, we prove Yao's observation [Yao82] about the equivalence of indistinguishability and hardness mentioned in Section 2.2. In the following Lemma, denote sampling a string uniformly at random from \( \{0,1\}^{n+1} \) by \( U_{n+1} \), and let \( U_n f(U_n) \) denote sampling \( x \) uniformly at random from \( \{0,1\}^n \) and outputting \( x f(x) \). Also, let \( \text{prefix}_n(x) \) denote the \( n \)-bit prefix of string \( x \).

**Lemma 52 ([Yao82])** For boolean function \( f : \{0,1\}^n \rightarrow \{0,1\} \), \( U_n f(U_n) \) is \( \epsilon \)-indistinguishable for size \( s \) from \( U_{n+1} \) if and only if \( f \) is \( (\frac{1}{2} - \epsilon) \)-hard for size \( s + \theta(1) \).

**Proof:**

- If \( f \) is \( (\frac{1}{2} - \epsilon) \)-hard for size \( s + \theta(1) \), then \( U_n f(U_n) \) is \( \epsilon \)-indistinguishable from \( U_{n+1} \) for size \( s \).

This proof is inspired by a proof in [KvM02]. Assume that \( U_n f(U_n) \) is \( \epsilon \)-distinguishable from \( U_{n+1} \) for size \( s \), then there exists some circuit \( C \) of size \( s \) that the following holds for it or its complement:

\[
\Pr[C(U_n f(U_n)) = 1] - \Pr[C(U_n U_1) = 1] > \epsilon
\]

Using this distinguisher one can build a probabilistic circuit \( C' \) that with equal probability simulates \( C'_0 \) or \( C'_1 \) of size \( s + \theta(1) \) to predict function \( f \) with probability greater than \( \epsilon + \frac{1}{2} \) as follows:
APPENDIX A. AUXILIARY PROOFS

\[ C'_r(x) = \begin{cases} r & C(xr) = 1 \\ 1 - r & \text{otherwise} \end{cases} \]

Namely, \( C'_r(x) = C(xr) \oplus r \). Now, partition \( \{0,1\}^n \) into \( X_1 \) and \( X_2 \), and consider the following two cases:

- **Case 1:** \( x \in X_1 \) such that \( C(xr) = C(x\bar{r}) \)
  
  Informally, choice \( r \) is irrelevant to the outcome of \( C \) in this case, and so
  
  \[ \Pr_{x \in U, X_1} [C(xf(x)) = 1] = \Pr_{x \in U, X_1, r \in U_1} [C(xr) = 1]. \]
  
  More precisely, in this case \( C'(x) \) (which is a boolean circuit) is a random bit, and without loss of generality:
  
  \[ \Pr_{x \in U, X_1} [C'_0(x) = f(x)] = \frac{1}{2} = \frac{1}{2} + \Pr [C(xf(x)) = 1] - \Pr [C(y) = 1] \]

- **Case 2:** \( x \in X_2 \) such that \( C(xr) \neq C(x\bar{r}) \)
  
  This implies that \( C'_0(x) = C'_1(x) \). Then, \( \Pr_{x \in U, X_2, r \in U_1} [C(xr) = 1] = \frac{1}{2} \), and \( C'(x) = f(x) \) iff \( C(xf(x)) = f(x) \). Thus, the following holds:
  
  \[ \Pr_{x \in U, X_2} [C'(x) = f(x)] - \frac{1}{2} = \Pr_{x \in U, X_2} [C(xf(x)) = 1] - \Pr_{x \in U, X_2, r \in U_1} [C(xr) = 1] \]

Thus, there exists \( b \in \{0,1\} \) such that:

\[
\Pr \left[ C'(x) = f(x) \right] = \Pr \left[ x \in U \right] \Pr_{x \in U, X_1} \left[ C'(x) = f(x) \right] + \Pr \left[ x \in U \right] \Pr_{x \in U, X_2} \left[ C'(x) = f(x) \right]
\]

\[
= \frac{1}{2} + \Pr \left[ C(xf(x)) = 1 \right] - \Pr \left[ y \in U_{n+1} \right] \left[ C(y) = 1 \right] > \frac{1}{2} + \epsilon
\]

This shows that there exists a predicting circuit \( C' \) of size \( s' = s + \theta(1) \) that correctly computes function \( f \) with probability greater than \( \frac{1}{2} + \epsilon \) which is a contraction because \( f \) is an \((\frac{1}{2} - \epsilon)\)-hard function for size \( s' \).

- **If** \( U_nf(U_n) \) **is \( \epsilon \)-indistinguishable from** \( U_{n+1} \) **for size** \( s \), **then** \( f \) **is** \((\frac{1}{2} - \epsilon)\)-hard **for size** \( s + \theta(1) \).

We prove the contrapositive, and assume that there exists a circuit \( C \) of size \( s' = s + \theta(1) \) that correctly computes \( f \):

\[ \Pr_{x \in U_n} [C(x) = f(x)] > \frac{1}{2} + \epsilon. \]
Construct circuit $D$ as follows:

- **Given:** $y \in \{0,1\}^{n+1}$
- **Procedure:**
  
  Compute $C(\text{prefix}_n(y))$.
  
  If $C(\text{prefix}_n(y)) =$ last bit of $y$, then output 1.
  
  Else, output 0.

If $D$ is given $U_nf(U_n)$, then with probability greater than $\frac{1}{2} + \epsilon$, $D$ outputs 1. If $D$ is given a string from $U_{n+1}$, then probability of $D$ outputting a 1 is a random guess (i.e., $\frac{1}{2}$). As a result,

$$\Pr[D(U_n f(U_n)) = 1] - \Pr[D(U_{n+1}) = 1] > \frac{1}{2} + \epsilon - \frac{1}{2} = \epsilon$$

In other words, if there exists a circuit of size $s'$ that correctly computes $f(x)$ with probability more than $\frac{1}{2}$, then a distinguishing circuit can be constructed that shows that $U_n f(U_n)$ and $U_{n+1}$ are $\epsilon$- distinguishable.
Bibliography


<table>
<thead>
<tr>
<th>Reference</th>
<th>Title and Authors</th>
</tr>
</thead>
</table>