Constructions, Lower Bounds, and New Directions in Cryptography and Computational Complexity

by

Periklis A. Papakonstantinou

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Constructions, Lower Bounds, and New Directions in
Cryptography and Computational Complexity

Periklis A. Papakonstantinou
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2010

In the first part of the thesis we show black-box separations in public and private-key cryptography. Our main result answers in the negative the question of whether we can base Identity Based Encryption (IBE) on Trapdoor Permutations. Furthermore, we make progress towards the black-box separation of IBE from the Decisional Diffie-Hellman assumption. We also show the necessity of adaptivity when querying one-way permutations to construct pseudorandom generators á la Goldreich-Levin; an issue related to streaming models for cryptography.

In the second part we introduce streaming techniques in understanding randomness in efficient computation, proving lower bounds for efficiently computable problems, and in computing cryptographic primitives.

We observe [Coo71] that logarithmic space-bounded Turing Machines, equipped with an unbounded stack, henceforth called Stack Machines, together with an external random tape of polynomial length characterize $\mathsf{RP}, \mathsf{BPP}$ an so on. By parametrizing on the number of passes over the random tape we provide a technical perspective bringing together Streaming, Derandomization, and older works in Stack Machines. Our technical developments relate this new model with previous works in derandomization. For example, we show that to derandomize parts of BPP it is in some sense sufficient to derandomize BPNC (a class believed to be much lower than $\mathsf{P} \subseteq \mathsf{BPP}$). We also obtain a number of results for variants of the main model, regarding e.g. the fooling power of Nisan's pseudorandom generator (PRG) [Nis92] for the derandomization of BPNC^1, and the relation of parametrized access to NP-witnesses with width-parametrizations of SAT.

A substantial contribution regards a streaming approach to lower bounds for problems in

the NC-hierarchy (and above). We apply Communication Complexity to show a streaming lower bound for a model with an unbounded (free-to-access) pushdown storage. In particular, we obtain a $n^{\Omega(1)}$ lower bound simultaneously in the space and in the number of passes over the input, for a variant of inner product. This is the first lower bound for machines that correspond to poly-size circuits, can do PARITY, Barrington's language, and decide problems in $\mathsf{P} - \mathsf{NC}$ assuming $\mathsf{EXP} \neq \mathsf{PSPACE}$.

Finally, we initiate the study of log-space streaming computation of cryptographic primitives. We observe that the work on Cryptography in NC^0 [AIK06a] yields a non-black-box construction of a one-way function computable in an $O(\log n)$-space bounded streaming model. Also, we show that relying on this work is in some sense necessary.

Acknowledgements

It is a great privilege to study Theory at the University of Toronto. I was also privileged to have an incredible advisor and PhD committee. I'm most thankful to Charles Rackoff for his invaluable research supervision, countless discussions, very close attention to my work, and insightful comments. In addition, Charlie is among the most open and liberal-minded individuals I have met in academia, which made this process more interesting. I'm also grateful to the other members of my committee Allan Borodin, Stephen Cook, and Toniann Pitassi for always been there with useful remarks, research suggestions, and encouragement. Many thanks to my external thesis reviewer Eric Allender for his detailed remarks and suggestions.

Significant part of the research I have included in the thesis is a result of collaboration, mostly with people from the University of Toronto. I want to thank my office-mates and colleagues, and co-authors (from UofT and elsewhere) Dan Boneh, Mark Braverman, Yuval Filmus, Michail Flouris, Travis Gagie, Wesley George, Victor Glazer, Costis Georgiou, Kaveh Ghasemloo, Hamed Hatami, Ali Juma, Lap Chi Lau, Dai Le, Tsuyoshi Morioka, Phuong Nguyen, Yevgeniy Vahlis, Brent Waters, and Anastasios Zouzias. Special thanks are due to my friends and co-authors Matei David and Anastasios Sidiropoulos.

I'd like to thank the faculty from the Department of Mathematics and in particular Péte Gabor, Dror Bar-Natan, and Bálasz Szegedy for widening my view on mathematics.

Let me also thank Stavros Cosmadakis for his influence during my undergraduate studies.

Finally, I want to thank my parents Antonios and Rania, my dear sister Danai, and my life-lasting friends Jenny Bay, Thanos Tsiantoulis and especially Yiorgos Peristeris, whose unconditional support and occasional intervention made this project manageable. I reserved the last place for a special person who makes my life brighter, meaningful and challenging in many ways. This is the wonderful Nan Zhou, my partner, friend, and lover. Thank you Nan.

Contents

Definitions and notation

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 1

Introduction

Randomness and computational intractability are key concepts in Cryptography and Computational Complexity that constitute the general theme of this thesis. This work goes in two main directions. The first regards *black-box separations* and *non-black-box constructions* in Cryptography. The second refers to two new approaches, orthogonal to every previous approach, in the study of fundamental Computational Complexity questions; one in understanding the *role of randomness* in efficient computation, and the other in proving *resource lower bounds*. The conceptual link between the main directions is streaming models and techniques.

This work proposes new definitions and approaches to fundamental open questions, and it exploits basic properties of these approaches. Occasionally there are involved technical steps, and it is interesting that the corresponding seemingly unrelated areas blend well together. Identifying connections between these areas is part of the contribution. The study is comprehensive in the sense that the statements cannot be improved modulo the techniques that are employed.

In the remainder of this chapter we outline the main parts of the contribution and their connections, and we provide necessary definitions and references to previous work.

1.1 Black-box separations in Cryptography

Most common cryptographic primitives and protocols base their construction in a *black-box* manner on assumptions about the existence of other primitives; e.g. Public Key Encryption based on the existence of trapdoor permutations [Yao82], and the Cramer-Shoup cryptosystem [CS98] based on the hardness of the Decisional Diffie Hellman (DDH) problem. A construction is *black-box* if it is based on a primitive in a way such that no explicit implementation of the primitive is needed; instead, the primitive is realized as an oracle. In Chapter 2 we show that certain primitives from Public and Private Key Cryptography cannot be based in a black-box way on certain popular primitives.

As cryptographic tasks become more complex, new and stronger assumptions are used in cryptographic constructions. A central question in Cryptography is whether these tasks can be based on more well-studied and weaker assumptions. It was an open question whether Identity Based Encryption (IBE) - a generalization of public key encryption - can be based on the popular assumptions regarding the existence of Trapdoor Permutations (TDPs) and the hardness of the Decisional Diffie-Hellman (DDH). In joint work with Boneh, Rackoff, Vahlis, and Waters [BPR+08] we answer the question for TDPs in the negative, in the sense that IBE cannot be based in a *black-box* way on TDPs. We have made substantial progress on the black-box separation of IBE from the DDH, and we present the details of a central technical difference between this work and [BPR+08].

To make sense of a precise statement of such theorems we first need to specify the setting where the black-box separation is shown. In Section 1.1.1 we present the setting for TDPs. The setting for DDH is interesting on its own right and we defer its presentation and discussion to Chapter 2.

In the second part of Chapter 2 we give a simpler black-box separation which refers to a private-key setting. In joint work with Juma [JP10] we consider Goldreich-Levin-like constructions of super-linear pseudorandom generators from one-way permutations. We show that composing the permutation with itself - i.e. making adaptive use of the permutation - is necessary in some sense. In fact, we prove a more general statement showing that (in a black-box setting) adaptivity is necessary when constructing pseudorandom generators. Understanding the role of adaptivity in cryptographic constructions is a self-motivated topic. However, our main motivation comes from the fact that non-adaptive constructions are necessary for building *Streaming Models for Cryptography* (Chapter 5), where the working memory and the ability to reread the seed are both very restricted.

1.1.1 Black-box separation of IBE from TDPs

Previous work and the main question. Identity-Based Encryption (IBE) [Sha85] is a public key system where any string can be used as a public key. Public keys in an IBE are often called identities. IBE systems have numerous applications in Cryptography; e.g. [CHK03, BCHK06, DF02, YFDL04, BCOP04, Lys02, BGW05, BDS+03] to name a few. Although the IBE concept was introduced in 1984 it took many years until the first practical constructions appeared in 2001 [BF03, Coc01]. We would say that there are four kinds of popular, well-studied assumptions on which modern public-key cryptography is based on: (i) TDPs, (ii) DDH, (iii) quadratic residues, and (iv) lattice assumptions. To date most constructions of Identity Based Encryption are based on bilinear pairings (generalization of DDH), with the exceptions being [BGH07, Coc01] who construct an IBE based on the quadratic residues assumption (in a random

oracle setting), and [GPV08] who construct an IBE based on lattices. It was open whether it is possible to base IBE constructions on (i) or (ii) in a black-box setting.

A TDP is a permutation that it is easy to compute, but hard to invert on the average without some additional information (the trapdoor). A TDP is a primitive stronger than that of one-way permutations (OWPs), where we require that the permutation is easy to compute and hard to invert (see Definition 1.1.2 below). TDPs can be used as a basis for semantically secure public-key encryption (e.g. [Yao82]), whereas the same is not true for OWPs, at least in a black-box manner [IR88].

Here is a definition of TDPs. We present a minimally general form of the definition which we believe immediately introduces the main concept. For a more careful and general treatment see e.g. [Gol01] (p.58).

Definition 1.1.1. A *Trapdoor one-way Permutation* (TPD) is a triple of polytime algorithms (G, E, D), where for every $n \in \mathbb{Z}^+$

- $G : \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$, on randomness r, $G(r) = (\mathsf{pk}, \mathsf{sk})$. We call G the *key-generation algorithm*, and pk,sk the public and the secret (private) key respectively.
- $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, such that for every $\mathsf{pk} \in \{0,1\}^n$, the restriction of E, $E(\mathsf{pk}, \cdot) : \{0,1\}^n \to \{0,1\}^n$ is a permutation. We call E the *encryption* algorithm.
- $D : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, such that on input sk and y we have $D(\mathsf{sk}, y) = x$, given that for some r, $G(r) = (\mathsf{pk}, \mathsf{sk})$, and $y = E(\mathsf{pk}, x)$. We call D the *decryption* algorithm.

and (G, E, D) has the following security property: For every polynomial time probabilistic algorithm Adv and every $k > 0$ and sufficiently large n such that

$$\mathsf{Pr}[\mathsf{Adv}(\mathsf{pk}, E(\mathsf{pk}, x)) = x] \leq n^{-k}$$

where the probability is over random $r, x \in \{0,1\}^n$, such that $(\mathsf{pk}, \mathsf{sk}) \leftarrow G(r)$.

Note that although it seems less natural, we could have instead of sk used the randomness r for G as a private key.

Here is the main intuitive question motivating this part of the research.

> *Can we, and in what sense, establish a negative result for basing, in a black-box way, an IBE system on TDPs?*

Initiated by [IR88], it is common for a black-box separation to be performed constructively in an information theoretic setting (or with access to a PSPACE oracle). Typically, the algorithms and the adversaries are computationally unlimited, and they are given access to an oracle

realizing the primitive; the resource we measure is the number of oracle queries. In the literature there are numerous, e.g. [Sim98, KST99, GGKT05, GKM+00, GMM07, BMG07, HHRS07], black-box separation results, though not all of them fall into the same technical template.

For a taxonomy of black-box separations, intuitive interpretations, and philosophical discussions see [RTV04].[1] Let us be more explicit on one interpretation. We know that IBEs exist under certain computational assumptions. Hence, in the study of the impossibility of constructing IBEs these assumptions must be disabled. A clean way of doing so is letting the algorithms be computationally unlimited. At the same time, since we are studying black-box constructions based on TDPs we provide the functionality of TDPs, but not any particular implementation, though oracle access. This oracle should realize a primitive where TDPs are feasible, and furthermore TDPs are naturally related to the primitive in the oracle.

Oracle setting. An IBE system (see below) consists of four algorithms. For the black-box separation we consider algorithms that have access to the same three random oracles g, e, d that roughly correspond to key generation, encryption, and decryption oracles of the TDP. We use $\mathcal{O} = (g, e, d)$ to refer to the triple of oracles. For every security parameter $\lambda \in \mathbb{Z}^+$, the oracles g, e, d are sampled uniformly at random from the set of all functions satisfying the following conditions:

- $g : \{0,1\}^\lambda \to \{0,1\}^\lambda$. We view g as taking a secret key sk as input and outputting a public key pk.
- $e : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a function that on input $\mathrm{pk} \in \{0,1\}^\lambda$ and $x \in \{0,1\}^\lambda$ outputs $e(\mathrm{pk}, x) \in \{0,1\}^\lambda$. We require that for every $\mathrm{pk} \in \{0,1\}^\lambda$ the function $e(\mathrm{pk}, \cdot)$ be a permutation of $\{0,1\}^\lambda$.
- $d : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a function that on input $\mathrm{sk} \in \{0,1\}^\lambda$ and $y \in \{0,1\}^\lambda$ outputs an $x \in \{0,1\}^\lambda$ that is the (unique) pre-image of y under the permutation defined by the function $e(g(\mathrm{sk}), \cdot)$.

We note that there is a natural uniform distribution defined over the finite set of all triples (g, e, d) satisfying these conditions. Also note that our oracles model well-known, real-world trapdoor permutations (e.g. RSA).

Informal description of an IBE and statement of the main theorem. We defer the precise definition of IBE and the definition of security to the relevant Chapter 2. Informally, an IBE system consists of four algorithms: *Setup, Key-generation, Encoding,* and *Decoding*

[1] Under this classification, we show that there is no *fully black-box reduction* between the relevant primitives.

algorithm. The Setup algorithm creates a master public MPUB and master private/secret MSK key. Every algorithm has (potentially) access to MPUB, but only the Key-generation algorithm accesses the MSK. The Key-generation algorithm issues for every identity ID a secret key $\mathsf{SK}_{\mathsf{ID}}$, which is the private key associated with the particular identity. Encryption is being done by using as a key the master public key together with the identity, whereas decryption is being done using the $\mathsf{SK}_{\mathsf{ID}}$. Regarding the security definition, for the moment let us just mention that it is a weak form of *semantic security*, and since we are showing a lower bound the weaker the security definition the stronger the result is. We are ready to give an informal statement of our main theorem.

Theorem (informal). Consider an IBE system where all four algorithms have access to the same randomly chosen oracle $\mathcal{O} = (g, e, d)$ and each of them makes at most α queries to the oracle. Then, there exists an adversary Adv, generic algorithm with oracle access to σ, which breaks the security of the IBE with $\alpha^{O(1)}$ many queries.

The black-box separation theorem follows from the above theorem together with the observation that a randomly chosen $\mathcal{O} = (g, d, e)$ satisfies Definition 1.1.1 with high probability. Let us also remark that IBE is a special case of *Functional Encryption*, and as such our separation result applies to this too.

In [PRVW10] (work still in progress) we are separating IBE from DDH. This separation is significantly more technical than the one in [BPR$^+$08]. The relativized setting is that of *Generic Groups Model* [Sho97]. This would be the first negative result of this type. Note that although this model has been criticized, perhaps not unfairly, e.g. [KM07] as too strong to provide meaningful positive results, we show a negative result even in this model. In Chapter 2 we discuss the model, and we technically deal with the main difference compared to the argument in [BPR$^+$08], by reducing the security of the general IBE to a restricted IBE system.

1.1.2 On the necessity of adaptivity in Goldreich-Levin

Our second type of black-box separation regards the construction of cryptographic[2] pseudorandom generators PRGs of super-linear stretch based on the Goldreich-Levin (GL) construction from one-way permutations [GL89]. In particular, we show that in the standard way of using GL to construct a pseudorandom generator with polynomial stretch, if we determine the queries to the OWP π non-adaptively (without accessing π) instead of composing π with itself, then we can break the security of this construction. We remark that the IBE black-box separation is technically complex, whereas this second black-box separation is less technical.

[2]In the thesis we consider two types of pseudorandom generators: for (i) cryptography and for (ii) derandomization. When necessary we emphasize by writing "cryptographic PRG" to distinguish.

Definition 1.1.2. A *one-way function (OWF)* $f : \{0,1\}^* \to \{0,1\}^*$ is a polytime computable function with the following security property. For every randomized polytime algorithm Adv, for every k and sufficiently large n

$$\Pr_{r \in U_n} [\mathsf{Adv}(1^n, f(r)) \in f^{-1}(f(r))] \leq n^{-k}$$

where the probability is over r and the randomness of Adv.

A *cryptographic pseudorandom generator* PRG G is a polytime computable function such that there is $s(n) \geq 1$, the *stretch* of the PRG, where for every $n \in \mathbb{Z}^+$, when G is restricted to inputs of length n, $G|_n : \{0,1\}^n \to \{0,1\}^{n+s(n)}$. In addition, G has the following security property. For every randomized polytime algorithm Adv, for every k and sufficiently large n

$$|\Pr_{r \in U_n} (\mathsf{Adv}(G(r)) = 1) - \Pr_{r \in U_{n+s(n)}} (\mathsf{Adv}(r) = 1)| \leq n^{-k}$$

We similarly define the non-uniform security versions where the adversary Adv instead of being a polytime Probabilistic Turing Machine (PTM) is a family of polysize (polynomial size) circuits.

Let $\pi = \{\pi_n : \{0,1\}^n \to \{0,1\}^n\}_n$ be a one-way permutation. Then, the standard way (e.g. [Gol01] p.128) of constructing a PRG G of polynomial stretch is the following. Let (r, x), $|x| = |r|$ be the input seed.

$$(r, x) \mapsto r, \langle r, x\rangle, \langle r, \pi(x)\rangle, \langle r, \pi^2(x)\rangle, \ldots, \langle r, \pi^{n^\alpha}(x)\rangle \tag{1.1.1}$$

where $\pi^i := \underbrace{\pi \circ \pi \circ \ldots \circ \pi}_{i \text{ times}}$, and $\langle \alpha, \beta \rangle$ denotes the standard inner product over $\mathsf{GF}(2)$.

We ask whether composing π with itself is necessary, in the sense that there exists a function $f : \{1, 2, \ldots, n^\alpha\} \times \{0,1\}^n \to \{0,1\}^n$ which determines the queries to π in a way independent to π, such that the following is pseudorandom.

$$(r, x) \mapsto r, \langle r, x\rangle, \langle r, \pi\big(f(1, x)\big)\rangle, \langle r, \pi\big(f(2, x)\big)\rangle, \ldots, , \langle r, \pi\big(f(n^\alpha, x)\big)\rangle \tag{1.1.2}$$

Theorem. *There exists a relativized setting where π is a OWP and the function defined in (1.1.2) is not pseudorandom.*

In fact, we prove this theorem in a more general setting (Theorem 2.3.2), where instead of inner product we have an arbitrary boolean function.

1.2 Parametrizing access to auxiliary memory: An approach to randomness

Chapter 3 deals at large with parametrizations of the auxiliary (random or non-deterministic) tape in *space*-bounded computation. It consists of two parts. In the first, which is the main

part, we initiate the study of a new approach to randomness in polynomial *time* computation by quantifying the number of times random bits are accessed. Defining the concept is the main contribution, and the technical developments relate this definition with previous questions in efficient probabilistic computation. We stress that our interest in this definition is not about the concept itself, but rather regarding its *technical* implications and potential. The second part of Chapter 3 deals with variants of the model. Certain success in these variants serves as indication that similar techniques may be applicable to our original model studied in the first part. Furthermore, these variants are of independent interest. For example, using a simple argument we make progress regarding the status of $\mathsf{PolyLogSpace} \not\subseteq \mathsf{P}$, for which nothing was known before.

Part of the content of Chapter 3 appears in [Pap09], and in joint work with David and Sidiropoulos [DPS09b, DPS09a].

Randomness is central to computer science and its intersection with engineering, and natural sciences. Some of the most intriguing questions in Computational Complexity regard the gap between polynomial time and its probabilistic analogs, e.g. RP, BPP. It is conjectured (e.g. [IW97, KI04, NW88]) that this gap is small or even that randomness is in general not necessary. Derandomizing probabilistic polynomial time (e.g. $\mathsf{P} = \mathsf{BPP}$) is a difficult mathematical problem since it is ultimately related to proving superpolynomial circuit lower bounds [KI04].

Consider a randomized procedure performing a random walk in a graph. After each transition the algorithm "forgets" the random bits used so far. Contrast this with a randomized procedure where random bits used at one step of the computation are revisited in future steps. We aim for a definition which allows us to distinguish between these two scenarios. Our contribution to the area is a new approach to quantifying restricted access to randomness during *arbitrary* polynomial time (polytime) computation. Unlike previous approaches which quantify the number of random bits, we parameterize the number of times the random tape is being scanned. The intention is to formalize the concept of "use of randomness in polytime computation". The way we formalize this intuition is somewhat debatable, given that such a definition is not meaningful for a regular polytime bounded Turing Machine (TM). A TM equipped with a read-only random tape in a single pass over the random tape can copy on its workspace all the random bits it's going to use. Hence, defining such a parametrization is a non-obvious task.

We apply a space-bounded characterization of polynomial time using *Stack Machines* (SMs) due to Cook [Coo71] (Cook calls these machines Auxiliary PushDown Automata). Identifying polynomial time as logspace computation equipped with unbounded recursion, is one of the most intriguing, albeit nowadays non-central, concepts in Theoretical Computer Science. We extend Stack Machines by adding a read-only, polynomially long random tape and we quantify on the number of passes over the random tape. All Stack Machines have a read-only input

tape, and logarithmic worktape unless mentioned otherwise. We refer to a Stack Machine with an auxiliary (e.g. random or non-deterministic) tape as a *Verifier Stack Machine* or vSM.

Definition 1.2.1. For a language $L \subseteq \{0,1\}^*$ we say that $L \in \mathsf{RPdL}[r]$ if there exists a vSM M such that for every $x \in \{0,1\}^*$ it makes at most $r(|x|)$ passes (i.e. it reverses $r(|x|) - 1$ times) over its polynomially long, read-only random tape, and if $x \in L$ then $M(x)$ accepts with probability $\geq \frac{1}{2}$, else if $x \notin L$ then $M(x)$ rejects with probability 1.

RPdL stands for Randomized Push-down Logspace. This abbreviation is chosen to emphasize the relation with RP, since RPdL with an unbounded (or exponential - see e.g. [Ruz81]) number of passes is RP.

By changing the error condition to be two-sided we similarly define $\mathsf{BPPdL}[\cdot]$ and $\mathsf{PPdL}[\cdot]$, whereas if we consider one-sided unbounded error (equivalently: the external tape contains non-deterministic bits) we define $\mathsf{NPdL}[\cdot]$.

We recall the definitions of probabilistic polynomial time e.g. $\mathsf{RP}, \mathsf{coRP}, \mathsf{BPP}$ in Chapter 3. For detailed definitions for probabilistic Turing Machines (PTMs), related classes, and introductory treatment cf. [AB09].

On the naturalness of the definition. Our formalization of the "use of randomness" in polynomial time computation goes through a space-bounded model. Thus, certain skepticism arises whether this definition defines the concept in a natural way. Philosophy aside, the RPdL hierarchy is introduced in order to be collapsed. The hope is that by changing the perspective we may obtain additional *technical* insight towards showing $\mathsf{P} = \mathsf{BPP}$. Regardless of any agreement or disagreement on its naturalness, from a technical perspective alone this definition creates an exciting potential. It brings together tools developed in many excellent works in Streaming, Communication Complexity, Derandomization, and Complexity Theory related to Stack Machines (AuxPDAs).

Current stage of research - Motivating questions. At this stage we were not able to derandomize BPP. We have obtained partial results which (i) provide connections of this new concept to previously known concepts, and (ii) for variants of this model we obtain results indicating possible future partial derandomization of RP or BPP, along with developments of independent interest. Here is the list of the main questions motivating the technical development.

1. How does $\mathsf{RPdL}[\cdot]$ relate to previously studied topics in derandomization?

2. Observe that $\mathsf{RPdL}[0] = \mathsf{P}$ and $\mathsf{RPdL}[2^{n^{O(1)}}] = \mathsf{RP}$ [Ruz81]. Is it possible that other than

the coincidence for these two extremes, in the intermediate $\mathsf{RPdL}[r(n)]$ levels things don't behave smoothly as $r(n)$ grows from 0 to $2^{n^{O(1)}}$?

3. Is there any indication that this approach may yield universal types of derandomization that were previously unknown?

4. What can be said for variants of the model?

Questions (1) and (2) are addressed through the same technical results. Similarly, for questions (3) and (4) every indication and success in derandomization, through our approach, refers to variants of the model. In particular, we define similar hierarchies roughly between NC^1 and RNC^1 (and BPNC^1), and succeed in derandomizing the polylogarithmic range of this hierarchy. Also, for other variants of the model we provide some results of independent interest.

1.2.1 Related work

Stack Machines. [Coo71] characterizes polytime computation in terms of Stack Machines (AuxPDAs): logspace bounded TMs equipped with an unbounded stack. Recursive logspace computation is a natural concept. Moreover, SMs exhibit quite interesting equivalences to other models of computation. Our work relies on the connections between deterministic (and nondeterministic) logspace and time-bounded SMs, and that of SAC (and NC) circuits with various forms of uniformity [All89, BCD$^+$89, Ruz80, Ruz81, Ven91]. When simultaneously to the space, we also bound the time, there is a constructive and direct correspondence (which intermediately goes through ATMs) between SMs and combinatorial circuits. This enable us to blur the distinction between space-time bounded SMs and families of circuits. In a study of NC in presence of strong uniformity (P-uniformity) [All89] studies SMs with restricted input-head moves. This is related to the concept of head-reversals on the random tape considered in our paper. For example, the discussion in [All89] (p.919) about the fact that the natural restriction on the head-moves of a SM translates to something more subtle in case of ATMs, is related to our definition of essential use of randomness using SMs with restricted number of scans on the random tape.

Derandomization. There is a long line of research in derandomization. Successful stories range from straightforward algorithms e.g. for MaxSAT (e.g. [AB09]) to quite sophisticated ones such as the AKS primality test [AKS04]. There is a plethora of works that aim to derandomize probabilistic polynomial time using certain types of pseudorandom generators. The majority of them relate the problem of derandomization to the problem of proving lower bounds for the average-case complexity of one-way functions [Yao82], or to arbitrary hard functions (cf.

[NW88, IW97, STV99, Uma03]). The general theme of this research direction comes under the title "randomness-hardness tradeoffs". These works are in line with our intuition about the circuit complexity of certain problems in E, EXP or NEXP. They provide evidence that derandomization of classes such as BPP might be possible, and simultaneously they give evidence about the conceptual difficulty of achieving such a goal.

Accessing the random tape. To the best of our knowledge, prior to our work there are only few other works asking similar questions. A corollary of [KV85] is PSPACE can be characterized (with zero-error) with a logspace bounded TM with two-way access over an exponentially long read-only random tape. The work closest to ours is [Nis93a]. Nisan shows that a logspace TM with two-way access over a polynomially long random tape decides with zero-error the languages decided by such machines where the error is 2-sided and the random tape is being accessed once. In [Nis93a]'s notation $\mathsf{BPL} \subseteq \mathsf{ZP}^*\mathsf{L}$.

1.2.2 Contribution

Definitions of complexity classes are given in Chapter 3. Our exposition is RP-centric; everything holds for BPP, the 2-sided error case.

The main result of [Coo71] implies the first non-trivial derandomization of $\mathsf{RPdL}[1] = \mathsf{P}$ (i.e. a single pass over the randomness). In case of 2-sided error things are more complicated - we show $\mathsf{BPPdL}[1] = \mathsf{P}$ in Section 3.3.

The randomness complexity grows smoothly with the level of the hierarchy. We relate the question of derandomizing along the RPdL hierarchy to the question of derandomizing RNC. In particular, derandomization along the RPdL hierarchy implies derandomization of RNC. More importantly a partial converse, modulo the derandomization technique, holds. That is, if RNC is being derandomized using pseudorandom generators, then the same generators can be used to derandomized RPdL.

> *Derandomizing a class believed to be strictly inside P, we derandomize a class which contains P.*

Technically, we proceed by introducing[3] the model of *Randomness Compilers*, which is also of independent interest. A randomness compiler M is a polytime transducer that given an input x outputs a polysize circuit of depth polylogarithmic in $n = |x|$. Then, acceptance of x is being determined by the acceptance probability of this circuit when it is evaluated on random inputs.

[3]This model was suggested by Mark Braverman who asked about its relation to probabilistic vSMs.

If for a language L, M outputs polysize circuits of constant fan-in and depth $O(\log^i n)$ then we say that $L \in \mathsf{P}+\mathsf{RNC}^i$, where acceptance/rejection has one-sided error.

We remark that $\mathsf{P}+\mathsf{RNC}$ is different from P-uniform RNC, since in $\mathsf{P}+\mathsf{RNC}$ the circuit depends on the input, not only on its length. In particular, $\mathsf{P} \subseteq \mathsf{P}+\mathsf{RNC}$ whereas the same is believed not to be true if instead of $\mathsf{P}+\mathsf{RNC}$ we had RNC.

Our main theorem (Theorem 3.2.3) relates the RPdL to the $\mathsf{P}+\mathsf{RNC}$ hierarchy.

Theorem 3.2.3. Let $i \in \mathbb{Z}^+$. Then, $\mathsf{P}+\mathsf{RNC}^i \subseteq \mathsf{RPdL}[2^{O(\log^i n)}] \subseteq \mathsf{P}+\mathsf{RNC}^{i+1}$.

Remark 1.2.2. Consider the intuitive statement that polynomial size circuits of increasing depth correspond to increasing complexity. The fact that we compile all the randomness we need in such depth-growing polysize circuits (Theorem 3.2.3) makes precise the title "The randomness complexity grows smoothly with the level of the hierarchy".

A corollary of Theorem 3.2.3 is that to derandomize the quasipolynomially many (out of exponential) levels of RPdL it suffices to construct PRGs secure against RNC circuits. These PRGs are somewhat different from the cryptographic PRGs introduced earlier.

Definition 1.2.3 (PRGs against adversaries with fixed complexity bounds)**.** Let $0 < \epsilon < 1$, $k \geq 1$. Let $G : \{0,1\}^* \to \{0,1\}^*$ be a function, such that $G(z)$ is computable in time $2^{O(|z|^{1/k})}$. We say that G is an $(\mathsf{NC}^i, k, \epsilon)$-*pseudorandom generator* if for every non-uniform NC^i circuit-family $\mathcal{C}$ and for sufficiently large $|z| := n$, $z \in \{0,1\}^*$, where $|G(z)| = 2^{|z|^{1/k}} := m$

$$|\mathsf{Pr}_{z \in_R \{0,1\}^n}[C_m(G(z)) = 1] - \mathsf{Pr}_{\rho \in U_m}[C_m(\rho) = 1]| \leq \epsilon$$

where $C_m \in \mathcal{C}$ has m input bits.

The precise technical statement follows.

Theorem 3.2.4. Let $k, i \geq 1$. If there exists a $(\mathsf{NC}^{i+1}, k, \frac{1}{7})$-PRG, then $\mathsf{RPdL}[2^{O(\log^i n)}] \subseteq \mathsf{DTIME}(2^{O(\log^k n)})$.

Hence, if there exists a k for all i's then $\mathsf{RPdL}[n^{\mathsf{polylog}n}] \subseteq \mathsf{DTIME}(2^{O(\log^k n)})$, whereas if k is a function of i then $\mathsf{RPdL}[n^{\mathsf{polylog}n}] \subseteq \mathsf{QuasiP}$, where $\mathsf{RPdL}[n^{\mathsf{polylog}n}] := \cup_{k>0}\mathsf{RPdL}[2^{\log^k n}]$ and $\mathsf{QuasiP} := \cup_{k>0}\mathsf{DTIME}(2^{\log^k n})$. We remark that a slightly different argument proves that it is sufficient to assume that the PRG is secure against SAC^i circuits[4], instead of NC^{i+1}.

Variants of the model: concrete success stories. There are two kinds of variants of probabilistic vSMs. One where the interpretation of the auxiliary tape changes; e.g. it contains

[4]A family of SAC^i circuits is a family of AC^i circuits where the $\wedge$ gates have bounded fan-in and all the negations are to the input level.

non-deterministic bits. The second is where we get rid of the stack and instead we consider usual logspace machines together with a parametrized auxiliary tape, henceforth called vTMs. A vTM is a logspace bounded Turing Machine, equipped with a poly-long auxiliary tape on which we count the number of head-reversals. The first kind of variant leads to results that are conceptually of limited interest, so we will focus on vTMs. Among others, for vTMs we obtain results that serve as an indication that concrete results are possible in their stack-equipped counterpart.

The previous technical discussion (Theorems 3.2.3 and 3.2.4) on the derandomization of P+RNC leaves out a detailed treatment for P+RNC^1 using vSMs. In other words, it may be possible to derandomize the first few levels of RPdL without the full derandomization of RNC^2. In fact, we conjecture that the first, say polylogarithmically many levels are possible to derandomize without the need to resort to methods of much deeper theoretical understanding, other than the one the current state of knowledge suggests. To that end a generalization of classical space-bounded Nisan's PRG may be of use. Here is a similar discussion for such a derandomization for vTMs (i.e. machines without a stack).

Independent to our question, the derandomization of RNC^1 and BPNC^1 is by itself a major open question. It is easy to see that the simulation of NC^1 circuits by logspace machines, directly extends to the simulation of BPNC^1 circuits by probabilistic vTMs. Hence, to derandomize BPNC^1 it is sufficient to derandomize the computation of probabilistic vTMs. We make the discussion BPNC^1-centric, since for one-sided error using corollaries from Section 3.5.2 we can obtain part of the results without relying on pseudorandom generators.

By parametrizing on the number of passes over the random tape of a vTM we define a hierarchy of classes $\mathsf{BPL}[r(n)]$ (similar to $\mathsf{BPPdL}[r(n)]$), where $\mathsf{BPL}[0] = \mathsf{LogSpace}$, $\mathsf{BPL}[1] = \mathsf{BPL}$ and $\mathsf{BPL}[\mathsf{poly}] := \cup_{k>0}\mathsf{BPL}[n^k] \supseteq \mathsf{BPNC}^1$. Using a simple analysis of a more general property of Nisan's classical space PRG [Nis92] we obtain the following theorem. The same result can be obtained from [INW94], using a more complicated PRG. However, the fact that the original PRG of Nisan has the same property was not known before[5].

Theorem 3.4.5. $\mathsf{BPL}[\mathsf{polylog}] := \cup_{k>0}\mathsf{BPL}[\log^k n] \subseteq \mathsf{QuasiP}$.

We complement this result, by showing that the above statement roughly exhausts the fooling power of Nisan's PRG. That is, we provide a logspace distinguisher that breaks the PRG with polynomially many passes, even for seeds of size $2^{c\sqrt{\log n}}$, for every $c > 0$.

Finally, we consider nondeterministic vTMs, i.e. logspace Turing Machines, with a polynomially long, nondeterministic tape. We denote the corresponding class as $\mathsf{NL}[r(n)]$. Our study for these types of machines goes beyond the scope of derandomization, but it does fall

[5] Noam Nisan, personal communication.

into the same context of restricting access to the auxiliary tape. For the extreme $r(n)$'s we have that $\mathsf{NL}[0] = \mathsf{LogSpace}$, $\mathsf{NL}[1] = \mathsf{NL}$, and $\mathsf{NL}[\mathsf{poly}] = \mathsf{NP}$, since NP predicates can be verified in logspace. We are interested in understanding the effect of restricted access to NP-witnesses. We characterize the classes between NL and NP in a simple way that has several theoretical and practical implications. The practical implications are related to SAT-solvers. In particular, it is quite a coincidence that the class $\mathsf{NL}[\frac{w(n)}{\log n}]$ has as a complete problem the *pathwidth-parametrized satisfiability* problem of width $w(n)$. The relevant definitions including the definition of Exponential Time Hypothesis or ETH (see below) are given in Chapter 3. Here are two consequences of our study.

It is well-known[6] that $\mathsf{PolyLogSpace} \neq \mathsf{P}$, and the assumptions $\mathsf{PolyLogSpace} \not\subseteq \mathsf{P}$ and $\mathsf{P} \not\subseteq \mathsf{PolyLogSpace}$ are very well-known due to their *implications*. However, it was fundamentally unknown whether any known assumption implies e.g. $\mathsf{PolyLogSpace} \not\subseteq \mathsf{P}$. Here is a simple corollary of our study.

Theorem 3.5.8. $\mathsf{PolyLogSpace} \cap \mathsf{NP} \not\subseteq \mathsf{P}$, unless the ETH fails. Furthermore, under the same assumption there exists a naturally defined $L \in \mathsf{PolyLogSpace} \cap \mathsf{NP}$ and $L \notin \mathsf{P}$.

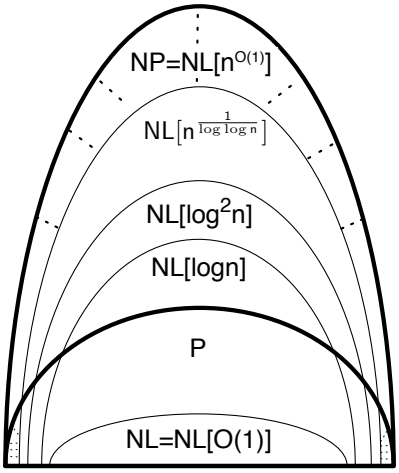
Under a further complexity assumption we show that the following diagram is correct. That is, the classes $\mathsf{NL}[r(n)]$, intersect P properly, all the way to the limit where we simultaneously get P and NP. For polylogarithmic $r(n)$ this makes them natural, syntactically[7] defined, classes between P and NP (and incomparable to P) closed under logspace reductions, with natural complete problems. In Chapter 5 we obtain an even stronger (and more complicated) result if instead ETH we have standard cryptographic assumptions (under the current state of knowledge incomparable to ETH).

1.3 An approach to lower bounds

Proving circuit lower bounds for efficiently computable problems is one of the main goals of Computational Complexity. In Chapter 4 we outline a new framework to obtain memory-access lower bounds *for problems in P, and in particular for problems in the NC hierarchy*, and we make the first non-trivial technical steps. The novelty of our general framework is that streaming lower bounds are translated to depth lower bounds for *polynomial size* circuits. We do this by observing a relation between older works in Computational Complexity for Stack Machines (AuxPDAs) and streaming on a space-bounded model with an *unbounded* stack. By unbounded we mean that we do not charge passes for accessing the stack. It is not obvious why

[6] The space hierarchy theorem implies that $\mathsf{PolyLogSpace}$ does not have complete sets under logspace reductions.

[7] Kaveh Ghasemloo made this observation.



any bounds are possible to be proved, since it is known that Stack Machines can perform useful P-computation (even outside NC !) by making a single pass over the input, and an exponential use of the stack [All89]. Compared to previous streaming models (i) we strengthen the model by introducing an *unbounded* memory tape and (ii) at the same time we restrict this memory to be a stack. We emphasize that with this modification technically different arguments (compared to usual streaming models) are required to obtain space-passes lower bounds. Stack Machines in Chapter 4 do not have an auxiliary tape and we quantify on the number of passes over the input tape.

A lower bound on the working space and the number of passes over the input can be independently interpreted as a streaming lower bound on a newly introduced model. Every such non-trivial bound involves technical novelty, at least when dealing with the *unboundness* of the stack. Therefore, we can obtain bounds on yet-another (impractical?) streaming model. However, apart from being a streaming model, logspace Stack Machines enjoy the fantastic property that they form a streaming model for languages in P, and in particular for $\mathsf{NC}^1, \mathsf{NC}^2, \ldots$. Conceptually, for the first time streaming lower bounds find applicability to the actual question about *polynomial size* circuits. The following theorem, corollary of [BCD$^+$89, Ruz80, Ruz81], makes this precise.

Theorem 1.3.1. *Let $\mathcal{C}$ be the class of languages decided by SMs that work in time $2^{O(\log^i n)}$. Then, $\mathsf{NC}^i \subseteq \mathcal{C} \subseteq \mathsf{SAC}^i \subseteq \mathsf{NC}^{i+1}$, where the circuit classes are (logspace) uniform. The same relation holds for non-uniform SMs and $\mathsf{NC}, \mathsf{SAC}$ circuits.*

This boils down to the following intuitive but precise statement.

> *Proving streaming lower bounds (on time or passes) for logspace Stack Machines means lower bounds for the depth of polysize circuits.*

We remark that under the streaming approach we obtain lower bounds for the number of passes, and in particular for time. However, passes are not the same as time. Logspace Stack Machines, as opposed to regular space-bounded models, under a widely believed assumption can decide sets in $\mathsf{P} - \mathsf{NC}$ with a single pass over the input.

1.3.1 Contribution

Using a Communication Complexity direct-sum type of argument in a joint work with David [DP10] we show a simultaneous $n^{\Omega(1)}$ space-passes (over the input) lower bound for every Stack Machine with an (unbounded) stack which solves P-EVAL, the problem of evaluating a polynomial over a field $\mathbb{F}$.

Problem: P-EVAL
Input: A point $y \in \mathbb{F}^n$, and polynomial $P \in \mathbb{F}[x_1, \ldots, x_n]$, where $\mathbb{F}$ is a fixed finite field..
Output: Evaluate $P(y)$.

Our main theorem follows by a Communication Complexity reduction.

Theorem (follows by Theorem 4.3.1). Let M be a SM for P-EVAL. Then, there are infinitely many $n \in \mathbb{Z}^+$ such that for each of them there exists $y \in \mathbb{F}^n$ where M makes at least n^ϵ passes over the input, for some $0 < \epsilon < 1$. This holds even for a fixed family of quadratic polynomials.

The above is a sublinear lower bound on the number of passes and in particular time. Thus, at this moment our lower bound is not as strong as to make direct use of Theorem 1.3.1, for obtaining circuit lower bounds for known classes. However, the machines for which we prove this theorem (i) can do PARITY and (ii) can decide sets in $\mathsf{P} - \mathsf{NC}$ (unless $\mathsf{EXP} = \mathsf{PSPACE}$), and thus regardless how the circuits characterizing the corresponding class look like, this lower bound corresponds to a new lower bound for some family of *polynomial size* circuits. We leave as future work the characterization of the circuit family for which our bound applies to.

Our techniques are new and we expect them to find broader applicability. The main issue is how to deal with the stack, whose presence significantly changes both the technical part of previous lower bound arguments, and also makes this model incomparable to other streaming models. These issues and the relevant model-separation statements are given in Chapter 4.

1.3.2 Related work

Communication Complexity and circuit lower bounds. Previous research relating circuit lower bounds and Communication Complexity introduced very interesting technical tools.

Karchmer and Wigderson [KW90] characterize precisely circuit depth in terms of a communication game for a language $L \subseteq \{0,1\}^*$. In particular, they show that the circuit depth required to decide L equals to the communication required - in the standard 2-party model [Yao79] - in the following game: let $x \in L$ and $y \notin L$, specify the bit where x and y differ. Such a game is known as KW-game. This application of Communication Complexity inspired a line of research, which resulted in *depth* lower bounds [KRW95, RW92, ERIS91], and finally the separation of the *monotone* NC hierarchy [RM97]. A different application of Communication Complexity, and in particular building upon the main result of [BNS92] for the NOF (Number On Forehead) model, Hastad and Goldman [HG91] obtain lower bounds for non-monotone depth-3 circuits with certain restrictions on the last level. Although, among this line of research there are approaches for non-monotone circuits (e.g. [KRW95] outlines an approach to separate NC^1 from AC^1), to date there has been no general approach relating Communication Complexity with generic *polysize* circuits.

Streaming lower bounds. There is a vast literature in streaming, too vast to survey here. Works related to sorting and frequency moment estimation somehow relate to our work. Perhaps the most relevant is the work in [GS05, BHN08], for which we provide a comparison in Chapter 4. The original streaming model of computation was introduced by [AMS99] (or in some sense even before [MP80]) in an attempt to model the huge real-world differences in access times to internal memory (e.g., RAM) and external memory (e.g., hard drives). In this model, the computation device is a space bounded TM that is allowed a bounded number of reversals on the input tape. There are a few extensions of this model, most relevant to us the one introduced by [GS05]. In this extension the streaming model of computation, called an (r,s,t)-read/write stream algorithm, is a TM with t external tapes on which the heads can reverse a total of r times, and internal tapes of total size s. Here is a problem that have attracted a few works in streaming: the *k-th frequency moment* F_k of a sequence $\alpha = (a_1, \ldots, a_M)$, $a_i \in [R]$ is $F_k(\alpha) = \sum_{j \in |R|} (f_j)^k$, where $f_j = |\{i \ : \ a_i = j\}|$. We refer the reader to [BHN08] for a short survey and references within. [BHN08], building on [BJR07, GS05, GHS06], shows that an $(o(\log n), s, O(1))$-r/w stream algorithm that approximates the k-th frequency moment of a data stream requires $s = n^{1-4/k-\delta}$ for any $k > 4$ and $\delta > 0$.

1.4 Some Remarks on Cryptography in Streaming Models

Chapter 5, the last part of the thesis, is the shortest in length and the technically simplest part. It contains a very simple non-black-box construction, which heavily relies on recently developed machinery [AIK06a] (and [HRV10]), and a few lower bounds indicating why relying

on this machinery is in some sense necessary.

Non-black-box techniques have recently gained importance over more conventional approaches in Cryptography. The resolution of the fundamental questions in Cryptography will involve non-black-box techniques with implications to landmark questions in Computational Complexity, such as $\mathsf{P} \neq \mathsf{NP}$. However, there are more modest goals addressed through non-black-box techniques, such as circumventing black-box separation results.

Chapter 5 investigates the possibility of streaming models for private key[8] Cryptography *from generic assumptions* [JP10]. Cryptography in Streaming models is motivated both from a theoretical and a practical point. The practical impact regards settings where on-line or streaming computation of the cryptographic primitive is needed. Theoretical motivation comes from the general theme of computing cryptographic primitives using rudimentary resources.

In Chapter 5 the machines are usual space-bounded machines without a stack. Here is the main motivating question for this part of the thesis.

> *Starting from generic assumptions, is it possible to implement a OWF using $O(\log n)$ space and a small ($= 1, 2, \ldots,$ constant, polylog) number of passes over the seed?*

We provide positive and negative results related to this question. The question itself can be answered in the positive for polylogarithmically-many passes. To that end, we construct a degenerate form of a very hard (but still subexponentially-hard) OWF, by a direct application of the main construction in [AIK06a].

Assumption 1 (Subexponentially-hard OWFs with parameter ϵ)**.** There exists a 2^{n^ϵ}-hard OWF f; i.e. f is hard to invert in time $\leq 2^{n^\epsilon}$ with probability $\geq 2^{-n^\epsilon}$, for sufficiently large input length n.

Throughout this chapter we insist on basing Cryptography in Streaming Models on generic assumptions such as the one above (similarly to the depth 4, NC^0 constructions in e.g. [AIK06b, AIK06a]). Apart from possible conceptual or practical implications of Cryptography in Streaming Models, we conjecture the following.

Conjecture. *There exists a 2^{n^ϵ}-hard OWF $\iff$ there exists a OWF computable in $O(\log n)$ space and $\log^{\epsilon'} n$ passes over the input seed.*

To the best of our knowledge there is no known non-trivial characterization of such very hard OWFs.

[8] This has no relation to, and should not be confused with "stream ciphers".

1.4.1 Contribution

Under a widely believed cryptographic assumption, we can easily establish the $\Rightarrow$ direction of this conjecture.

Theorem 5.2.3. Suppose that there exists a $2^{n^{\Omega(1)}}$-hard OWF, and a OWF computable in logspace (and no restriction on the number of passes over the input seed). Then, there exists a OWF computable in $O(\log n)$ space and $\log^{O(1)} n$ many passes over the input seed.

The existence of a OWF in logspace (with *unbounded many passes* over the input) is a very mild cryptographic assumption (see below on related work).

Note that the $O(\log n)$ space bound is essential for appreciating the issue. For example, consider a space bound $\omega(\log n)$ and a sufficiently hard OWF. Then, it is easy to see (Example 5.1.1) why there exists a OWF computable with a single pass over the input. The same is not true when the worktape becomes $O(\log n)$ small, even for arbitrary number of passes over the input. In fact, it is counter-intuitive why it is at all possible.

Also, we show a number of (unconditional) negative results illustrating possible obstacles for logspace streaming computation of a OWF. Using the last part of Chapter 3 we unconditionally obtain that no OWF computable with $O(1)$ many passes exists. Furthermore, we systematically study the necessity of a non-black-box construction of such a OWF. In particular, we show that natural implementations of popular, concrete intractability assumptions based on the hardness of FACTORING and SUBSET-SUM unconditionally require $n^{\Omega(1)}$ space and passes to compute. Then, we consider a possible black-box use of the NC^0 OWFs constructed in [AIK06b]. We show that NC^0 circuits (with multiple outputs) cannot, in general, be simulated by logspace transducers with a small number of passes. Finally, we show that there is no general simulation of an $\omega(\log n)$ space bounded transducer with a small number of passes over the input, by a $O(\log n)$ model with an at most polynomial overhead overhead in the number of passes over the input seed.

1.4.2 Related work

Streaming Cryptography falls into the general theme of computing cryptographic primitives using rudimentary resources, and dates back at least 20 years. The existence of PRGs in logspace (with an *unbounded number of passes* over the seed), or even AC^0 or TC^0, follows from standard intractability assumptions on the hardness of SUBSET-SUM, FACTORING, DISCRETE-LOG or lattice problems; e.g. [IN89, GL89, BM84, HILL99]. Regarding streaming constructions, where the number of passes over the input seed is bounded, Kharitonov et al [KGY89] study the possibility of such constructions, and they observe that $\omega(\log n)$ space is sufficient. For space $O(\log n)$, their main result is that no PRG of linear stretch exists when the input seed is

read a constant number of times. We improve on their main result by showing that no OWF (and in particular no PRG that stretches n bits to $n+1$) exists. Up until our work, for space $O(\log n)$, the literature deals only with impossibility results. Yu and Yung [YY94] show several lower bounds for types of automata and machines that work in sub-logarithmic space. For the next 15 years "rudimentary" means polynomial size circuits of constant depth and this line of work gives possibility results and lower bounds.

The major breakthrough regards the construction of OWFs and PRGs (and other primitives) in NC^0 and in particular for circuits of depth 3 and 4; note that Cryan and Miltersen[CM01] observe, based on the easiness of 2-SAT, that no PRG exists in NC^0 of depth 2. The seminal work of Applebaum, Ishai, and Kushilevitz [AIK06b, AIK06a, AIK08, AIK07] builds upon the work of Randomizing Polynomials e.g. [IK00], and shows the possibility of *Cryptography in* NC^0: given a "cryptographic function" f construct a randomized encoding of f, which is a distribution $\{\hat{f}\}$ that (i) preserves the security of f, and (ii) is much simpler to compute than f. In particular, starting from the generic assumption that "cryptographic" functions can be computed in logspace, the authors show how to construct a family of NC^0 circuits by appropriately compiling the corresponding polynomial size branching program. This amazing technical achievement brings the combinatorics of cryptographic functions to a simplified setting, and opens the possibility of better understanding cryptographic primitives. Our logspace streaming construction depends on the specifics of this work. We remark that our construction relies on circuits where every subcircuit is of small expansion. Goldreich [Gol00] proposes a OWF computable by NC^0 circuits of high expansion, and Bogdanov and Qiao [BQ09] compromise the security of this function for certain values of the parameters. Very recently, [HRV10] gives a highly parallel (and non-adaptive) construction of a PRG from an arbitrary OWF. In particular, this can be used to improve the [AIK06a] result by weakening a cryptographic assumption.

Chapter 2

Black-box separations in Cryptography

We answer in the negative two questions, one in Public-Key and one in Private-Key Cryptography. The first is whether we can base Identity Based Encryption (IBE) in a black-box setting on the existence of Trapdoor Permutations (TDPs) or on the average-case hardness of Decisional Diffie-Hellman (DDH). The second asks whether it is possible to base in a black-box and *non-adaptive* way the existence of pseudorandom generators (PRGs) on one-way permutations (OWPs).

For intuition, motivation, and preliminary definitions see Section 1.1. In Section 2.1 we show the impossibility of basing IBE on TDPs. The impossibility of IBE based on DDH is still work in progress. In Section 2.2 we define and discuss the relativized setting for the DDH separation, and we show that the security of a general IBE in this setting can be reduced to a restricted type of IBE. This is a central technical difference with the separation in Section 2.1. In Section 2.3 we show that it is not possible to construct in a black-box way PRGs of super-linear stretch, by making non-adaptive use of a one-way permutation (OWP). This black-box separation is conceptually related to Cryptography in Streaming Models (Chapter 5).

2.1 Black-box separation of IBE from TDPs

We ask whether a secure IBE can be built from the weaker primitive of TDPs using a black-box construction and without any further assumptions. We aim for a strong type of impossibility result. As such the question is whether we can construct a semantically secure IBE system for encrypting a single bit message. Here, we use a weakened version of IBE semantic security defined in [BF03]. In our definition the adversary non-adaptively requests private keys of identities of its choice even before seeing the public parameters, and then tries to break semantic

security for some other identity.

Our separation proof builds a natural oracle $\mathcal{O}$ relative to which TDPs exist. We then show that all candidate WSS-IBE systems $(\mathsf{Setup}^{\mathcal{O}}, \mathsf{Keygen}^{\mathcal{O}}, \mathsf{Enc}^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}})$ are insecure against a polynomial time adversary $\mathcal{A}$ with access to $\mathcal{O}$ and to a PSPACE oracle. The oracle $\mathcal{O}$ is very natural and directly implies the existence of trapdoor permutations. The difficult part of the proof is constructing the adversary Adv that breaks any given candidate SS-IBE system relative to this oracle.

Our proof brings to light the essential property of IBE systems: an IBE system creates an exponential number of public keys (identities) and compresses all of them into a short string called the public parameters. This ability to represent exponentially many public keys using a short string is not possible with a TDP. This exponential compression is achieved in existing IBE constructions using algebraic properties (see also Section 2.2) for further discussion.

For motivation and discussion see Section 1.1. In Section 2.1.1 we fix our notation, and define what is a *Weakly Semantically Secure IBE (WSS-IBE)* in the TDP Model. In Section 2.1.2 we state the black-box separation theorem and discuss its trivial direction, i.e. that in our oracle model TDPs exist. In Section 2.1.3 we state our main theorem and we describe the attack algorithm Adv. In Section 2.1.4 we give an outline of the proof, and Section 2.1.5 contains the details of the argument.

2.1.1 Definition of Weakly Semantically Secure IBE in the TDP Model

Recall that an IBE system consists of four deterministic algorithms [Sha85, BF01] $(\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$. In our case an IBE system also specifies constants $\lambda, q, n \in \mathbb{Z}^+$, and $\varepsilon \in (1/2, 1]$ as part of its description, an the algorithms have access to the oracles g, e, d described in Section 1.1 and is parameterized by λ. The IBE algorithms are allowed to make at most q queries to their oracle during a single execution. The parameter n is used to determine the number of random bits and the output lengths of the IBE algorithms. The parameter ε specifies the IBE's level of correctness, that is, a ciphertext generated from the correct distribution will be successfully decrypted with probability $\geq \varepsilon$.

The functionality of each algorithm is as follows: algorithm $\mathsf{Setup}(\mathsf{MSK})$ takes as input a master secret key $\mathsf{MSK} \in \{0,1\}^n$; it outputs public parameters $\mathrm{PP} \in \{0,1\}^n$. Algorithm $\mathsf{Keygen}(\mathsf{MSK}, \mathsf{ID})$ is deterministic, and it takes as input the master secret key MSK, and an identity $\mathsf{ID} \in \{0,1\}^n$. It outputs the private key $\mathsf{SK}_{\mathsf{ID}} \in \{0,1\}^n$ for identity ID. The encryption algorithm $\mathsf{Enc}(\mathrm{PP}, \mathsf{ID}, b, r)$ encrypts a bit b for a given identity ID using public parameters PP and randomness $r \in \{0,1\}^n$; it outputs a ciphertext $C \in \{0,1\}^n$. The decryption algorithm $\mathsf{Dec}(\mathsf{SK}_{\mathsf{ID}}, C)$ decrypts a ciphertext C using the private key $\mathsf{SK}_{\mathsf{ID}}$ and is deterministic. Note that making the algorithm Keygen deterministic does not restrict the type of IBE constructions

that our results cover since any secure IBE with a randomized Keygen can be converted into a secure IBE with a deterministic K by applying a pseudo-random function to the identity in order to obtain (pseudo) random bits for Keygen.

An IBE system must satisfy the following correctness property: for every $\mathsf{ID} \in \{0,1\}^n$, $b \in \{0,1\}$

$$\Pr\left[\begin{array}{l} \mathrm{PP} \leftarrow \mathsf{Setup}(\mathsf{MSK}),\ \mathsf{SK}_{\mathsf{ID}} \leftarrow \mathsf{Keygen}(\mathsf{MSK}, \mathsf{ID}), \\ C \leftarrow \mathsf{Enc}(\mathrm{PP}, \mathsf{ID}, b) \end{array} ;\ \mathsf{Dec}(\mathsf{SK}_{\mathsf{ID}}, C) = b\right] \geq \varepsilon$$

Normal Form. We restrict our attention (without loss of generality) to IBEs that satisfy the following condition: each oracle query of the form $d(\mathrm{sk}, \cdot)$ must be followed by the query $g(\mathrm{sk})$. We use this condition to avoid situations in which we know a trapdoor for some unknown trapdoor permutation.

Definition of Security. For security we use a weakened version of the IBE semantic security that is defined in [BF03]. The weak semantic security is defined using a game in which the attacker starts by choosing a polynomially long sequence of identities for which it wishes to obtain private keys, along with the challenge identity that it is going to try and break. This is done even before the adversary is given the public parameters. The adversary is then given the public parameters, a sequence of private keys for the above identities (except the challenge identity), and an encryption of a random bit for the challenge identity using the public parameters. More precisely, weak semantic IBE security is defined using a game between a challenger and an adversary Adv in which both parties are given a security parameter λ is input. The game proceeds as follows:

Setup: The adversary submits a challenge identity ID_*, and a sequence of identities $\mathsf{ID}_1, \ldots, \mathsf{ID}_l$ where $l = l(q)$ and $l(\cdot)$ is some polynomial.

Private Keys: The challenger chooses at random $\mathsf{MSK} \in \{0,1\}^n$. It then computes $\mathrm{PP} \leftarrow \mathsf{Setup}(\mathsf{MSK})$, and $\mathsf{SK}_i \leftarrow \mathsf{Keygen}(1^\lambda, \mathsf{MSK}, \mathsf{ID}_i)$ for $1 \leq i \leq l$. The tuple $(\mathrm{PP}, \mathsf{SK}_1, \ldots, \mathsf{SK}_l)$ is given to the adversary.

Challenge: The challenger chooses at random $r \in \{0,1\}^n$, $b \in \{0,1\}$. It then computes $C_* \leftarrow \mathsf{Enc}(\mathrm{PP}, \mathsf{ID}, b; r)$, and C_* is given to the adversary.

Guess: The adversary outputs a guess $b' \in \{0,1\}$, and wins if $b' = b$.

We define the advantage of the adversary Adv in attacking the scheme $\mathcal{E}$ as

$$\mathrm{Advantage}_{\mathcal{E},\mathsf{Adv}} := \left|\Pr[b = b'] - \frac{1}{2}\right|$$

The probability is over the random bits used by the challenger and the adversary.

2.1.2 The black-box separation Theorem

In the language of [RTV04] we show that there is no fully black-box reduction of the security of a TDP to a WSS-IBE. We do this by showing that in the relativized setting we defined above, TDPs exist (a.c. for a randomly chosen $\mathcal{O}$), and at the same time there exists an adversary that breaks the security of every (even the weakly semantically secure) IBE with access to $\mathcal{O}$. The first claim is immediate by the definition of the oracle setting. The other part is technically involved.

Claim 2.1.1. *There exists a trapdoor permutation scheme such that for all oracle adversary's $\mathcal{A}$ that make at most a polynomial number of queries to the oracle $\mathcal{O}$ we have that for sufficiently large λ the adversary's success in breaking the trapdoor permutation is negligible in λ, where λ is the security parameter of the scheme.*

Our main theorem proves something slightly more general than what mentioned above, since our attack is successful even against reasonably incorrect IBEs; i.e. as close to the correctness guarantee ε as we wish. Here is the main theorem.

Theorem 2.1.2. *Let $q \in \mathbb{Z}^+$, and $\varepsilon \in (1/2, 1]$. Then, for every IBE which makes at most q queries to the oracle, and has correctness parameter $\geq \varepsilon$, and for every $1/2 < \delta < \varepsilon$ there exists an adversary Adv and constant $c > 0$ such that Adv makes at most q^c oracle queries and decrypts the challenge ciphertext successfully with probability $\geq \delta$.*

2.1.3 The attack algorithm

In this section we describe an attack algorithm that breaks every IBE in the trapdoor permutation model. Our attack requires only a polynomial number of queries to the oracles, and can be implemented in polynomial time if we add a PSPACE oracle. We start with a high level overview of the attack.

The algorithm Setup in an IBE takes a random private MSK and outputs a public key PP. During its computation Setup may make use of its trapdoor permutation oracles, and use the results of the queries to construct the final public key. In our model any Setup algorithm that aims to be secure will have to make "essential" use of its oracles. Otherwise, an adversary could simply invert the function computed by Setup, thus obtaining a valid private key. Looking ahead to our analysis, we show that Setup has to make use of the trapdoor generation oracle Setup by generating several public keys of trapdoor permutations, and embedding them in the public parameters of the IBE. After the IBE public parameters are fixed the algorithm Keygen may include in each individual private key of an identity one or more trapdoors for the permutation public keys that were embedded in the IBE public parameters. We argue that

using the permutation public keys that are embedded in the IBE public parameters is the only way to encrypt securely. Thus, if the adversary could collect all the important[1] trapdoors for the embedded public keys it would be able to decrypt any ciphertext.

It is precisely this task of discovering the important trapdoors that is made possible due to the fact that the public parameters of an IBE encode an exponential number of public keys, and at the same time contain only a polynomial number of embedded permutation public keys. Suppose that a certain permutation, for which the public key is embedded in the public parameters, is used in a significant way to encrypt a bit to some identity. Then, for the decryption to work the corresponding trapdoor must be embedded in that identity's private key. Thus, if there are q permutation public keys embedded in the public parameters there can be at most q identities for which the identity's private key contains a trapdoor for one of the trapdoor permutations that are embedded in the public parameters. Thus, for every set of identities that is significantly bigger than q only a small portion of the identities have private keys that reveal new important trapdoors. This immediately yields the following approach for an attack: let S be a sufficiently large set of identities (say of size q^c). We select a challenge identity at random from S and ask to see the private keys for all the other identities in S. With probability $1 - 1/q^{c-1}$ the private keys that we obtained contain all the important trapdoors that are needed to decrypt ciphertexts that were addressed to any identity in S, including the challenge identity.

While the above argument gives an intuitive sketch for an attack algorithm, the actual algorithm and proof contain several challenging subtleties. The most interesting challenges arise from the fact that the public parameters and IBE private keys may be encoded in an arbitrary manner. In particular, an attack algorithm that simply "looks" at an IBE private key will not necessarily be able to tell which oracle secret keys are embedded in it. We resolve this issue by encrypting a random bit to that identity, and then decrypting the resulting ciphertext. We record all trapdoors that are actually used in queries to the permutation inverse oracle d during the computation of the encryption and decryption algorithms.

The fact that we discover important trapdoors by observing a sample computation of the encryption and decryption algorithms for each identity leads to the main technical complication in our attack. Since we discover only trapdoors that are used in the decryption of one ciphertext for each identity, it is crucial that all the values that appear during the decryption of the challenge ciphertext are distributed in a manner as they would be if the correct private key and the actual oracle were used for the decryption. Otherwise, if the decryption process for the

[1] Some trapdoor public keys may be generated by Setup and embedded in PP but never used by the encryption algorithm. For these public keys there may be no way of recovering the trapdoors with a small number of queries. However, it is also unnecessary as the adversary will never need these trapdoors to decrypt.

challenge identity looks significantly different from the decryption for the other identities then new trapdoors may be required to decrypt the challenge ciphertext.

Partial oracles. Our attack algorithm will generate during its computation trapdoor permutation oracles that are defined only on a subset of the possible inputs. We call such functions partial trapdoor permutation oracles. The following definition describes when such a partial oracle is extendable to a full oracle:

Definition 2.1.3. Let $\mathcal{O}' = (g', e', d')$ be a function which is defined on part of the domain of trapdoor permutation oracles. $\mathcal{O}'$ is called a *consistent partial oracle* if it has all the following properties:

(i) For every $\text{pk} \in \{0,1\}^\lambda$, $e'(\text{pk}, \cdot)$ is 1-1; (ii) for every $\text{sk} \in \{0,1\}^\lambda$, $d'(\text{sk}, \cdot)$ is 1-1; (iii) for every sk such that $g'(\text{sk})$ is defined let $\text{pk} = g'(\text{sk})$. We require that for every $x \in \{0,1\}^\lambda$, $e'(\text{pk}, x) = \alpha$ is defined if and only if $d'(\text{sk}, \alpha) = x$ is defined.

The Attack Algorithm

For notational simplicity we will occasionally refer to the algorithm without using the $\mathcal{O}$ superscripts.

Let $1/2 < \delta < \varepsilon \leq 1$, and $q, n \in \mathbb{Z}^+$. We now describe the attack algorithm Adv, which plays the Identity-Based Encryption security game with a challenger $\mathcal{C}$ for a given IBE system $IBE = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ which makes at most q queries, has inputs of length n, and has correctness guarantee $\geq \varepsilon$. Adv wins with probability $\geq \delta$.

The adversary starts by choosing five constants c_1, c_2, c_3, c_4, c_5 such that $c_4 \geq c_1 + 2$ and

$$\frac{2}{q^{c_3 - c_1 - 1}} + \frac{3}{eq^{c_2 - c_4}} + \frac{2}{q^{c_1 - 1}} + \frac{10}{q^{c_5 - (c_1 + c_2 + c_3 + c_4 + 2)}} \leq \frac{\varepsilon - \delta}{2}$$

Degenerate case

Our general adversary does not deal well with the following IBEs due to the small size of the probability spaces in question, therefore, we break such IBEs separately.

If $2^\lambda \leq q^{c_5}$ then the adversary proceeds as follows:

1. The adversary queries the oracles g, e, d on all 2^λ inputs. After this step the adversary has completely learned the oracle $\mathcal{O}$. Thus, from this point on the adversary will not make any additional oracle queries.

2. The adversary then chooses an arbitrary challenge identity $\mathsf{ID}_* = \bar{0}$ and asks to obtain an encryption C_* of a random bit to ID_*.

3. The adversary chooses MSK' such that $\mathsf{Setup}^{\mathcal{O}}(\mathsf{MSK}') = \mathrm{PP}$, and computes $\mathsf{SK}_* = \mathsf{Keygen}^{\mathcal{O}}(\mathsf{MSK}', \mathsf{ID}_*)$.

4. Finally, the adversary computes and outputs $\mathsf{Dec}^{\mathcal{O}}(\mathsf{SK}_*, C_*)$.

Clearly in the degenerate case the adversary successfully decrypts the ciphertext with probability ε.

The general case

If $2^\lambda > q^{c_5}$ our adversary performs the following steps.

Initialization. The adversary starts by initializing a table L, which will be used to store information that the adversary learns about the oracle $\mathcal{O}$. More precisely, L is a set of tuples of the form (α, β) where α specifies an oracle from $\{g, e, d\}$ and the query to that oracle, and β is the reply that was given to that query.

From this point on, every query that the adversary makes to the oracle $\mathcal{O}$, and the answer to that query are recorded in L.

Setup. Let $S \subseteq \{0,1\}^*$ be the set containing the binary encodings of the integers $1, \ldots, q^{c_1}$. The adversary picks uniformly at random $\mathsf{ID}_* \in_{\mathrm{R}} S$, submits ID_* as the challenge identity and requests to see private keys for all $\mathsf{ID} \neq \mathsf{ID}_*$, $\mathsf{ID} \in S$.

Challenge. The adversary is given a tuple $(\mathrm{PP}, \mathsf{SK}_1, \ldots, \mathsf{SK}_{\mathsf{ID}_*-1}, \mathsf{SK}_{\mathsf{ID}_*+1}, \ldots, \mathsf{SK}_{q^{c_1}}, C_*)$. From now on the adversary will make use of the public parameters PP and the private keys SK_i, but the challenge ciphertext C_* will only be used in the last stage of the attack.

Step 1: Discovering important trapdoors. For each $\mathsf{ID} \neq \mathsf{ID}_*$, $\mathsf{ID} \in S$, the adversary chooses at random $r \in_{\mathrm{R}} \{0,1\}^n$, $b \in_{\mathrm{R}} \{0,1\}$. Then, the adversary simulates $C \leftarrow \mathsf{Enc}^{\mathcal{O}}(\mathrm{PP}, \mathsf{ID}, b; r)$ and $\mathsf{Dec}^{\mathcal{O}}(\mathsf{SK}_{\mathsf{ID}}, C)$. Notice that during this step the adversary will add to L, among other things, mappings of the form $g(\mathrm{sk}) = \mathrm{pk}$. Whenever such a mapping is added to L we can invert the permutation pk successfully.

Step 2: Discovering frequent queries. The adversary repeats the following q^{c_2} times: chooses at random $r \in_{\mathrm{R}} \{0,1\}^n$, $b \in_{\mathrm{R}} \{0,1\}$, and computes $\mathsf{Enc}^{\mathcal{O}}(\mathrm{PP}, \mathsf{ID}_*, b; r)$.

Step 3: Discovering secret queries and decrypting the challenge. The adversary chooses at random $1 \leq k \leq q^{c_3}$ and repeats the following k times.

1. **(Offline phase)** The adversary chooses uniformly at random a global secret key MSK', and a partial oracle $\mathcal{O}' = (g', e', d')$ that satisfies the following properties: $\mathcal{O}' \setminus L$ is of

size[2] at most q^{c_4}; $L \subseteq \mathcal{O}'$, that is, $\mathcal{O}'$ contains the partial information that the adversary has learned about $\mathcal{O}$; $G^{\mathcal{O}'}(\mathsf{MSK}') = \mathrm{PP}$; For every $\mathsf{ID} \neq \mathsf{ID}_*$, $\mathsf{ID} \in S$ it holds that $\mathsf{Keygen}^{\mathcal{O}'}(\mathsf{MSK}', \mathsf{ID}) = \mathsf{SK}_{\mathsf{ID}}$; and $\mathsf{Keygen}^{\mathcal{O}'}(\mathsf{MSK}', \mathsf{ID}_*)$ is defined[3].

We also require that $\mathcal{O}'$ is a *consistent partial oracle* according to Definition 2.1.3.

2. Using the partial oracle $\mathcal{O}'$ the adversary computes $\mathsf{SK}'_* \leftarrow \mathsf{Keygen}^{\mathcal{O}'}(\mathsf{MSK}', \mathsf{ID}_*)$.

3. **(Online phase)** The adversary chooses at random $r \in_{\mathrm{R}} \{0,1\}^n$, $b \in_{\mathrm{R}} \{0,1\}$, and computes $C \leftarrow \mathsf{Enc}^{\mathcal{O}}(\mathrm{PP}, \mathsf{ID}_*, b; r)$. Notice that here we are using the actual oracle $\mathcal{O}$.

4. The adversary combines the partial oracle $\mathcal{O}'$ and the actual oracle $\mathcal{O}$ into a new oracle $\mathcal{O}''$. The precise description of $\mathcal{O}''$ is given in the next paragraph. The adversary then simulates $\mathsf{Dec}^{\mathcal{O}''}(\mathsf{SK}'_*, C)$. To decrypt the challenge ciphertext the adversary replaces C with C_* at the last repetition of this step. That is, at repetition k the adversary simulates $\mathsf{Dec}^{\mathcal{O}''}(\mathsf{SK}'_*, C_*)$ and obtains a bit b'.

Guess. The adversary outputs the bit b' obtained in the last repetition of the previous stage.

The oracle $\mathcal{O}''$

The oracle $\mathcal{O}''$ is constructed by the adversary to create a single oracle which combines the offline generated values of $\mathcal{O}'$ with the actual oracle $\mathcal{O}$. Note that we cannot avoid using the oracle $\mathcal{O}$ in the decryption since it was used to encrypt the challenge ciphertext.

To describe $\mathcal{O}''$ we need to treat specially a certain type of permutations. We call trapdoor permutations that are generated by Keygen for some ID, but not by Setup, "internal trapdoor permutation". Intuitively, the internal permutations are not accessible to the encryptor and therefore we do not have to use the actual oracle to answer queries about them. Formally,

Definition 2.1.4. Let $\mathcal{O}'$ be a partial oracle, and $\mathsf{MSK}' \in \{0,1\}^n$. We define

(i) U to be the set of pk such that $g(\mathrm{sk})$ is asked during $\mathsf{Setup}^{\mathcal{O}'}(\mathsf{MSK}')$ and received answer pk.

(ii) V be the set of pk such that $g(\mathrm{sk})$ is asked during $\mathsf{Keygen}^{\mathcal{O}'}(\mathsf{MSK}', \mathsf{ID})$
for some $\mathsf{ID} \in S$ and received answer pk.

(iii) L_g be the set of pk such that there exists a mapping of the form[4] $[g(\cdot) = \mathrm{pk}]$ in L.

The set of internal trapdoor permutations is $W = V \setminus U$.

[2] That is, the number of mappings that were invented by the adversary, and did not appear in L, is at most q^{c_4}.

[3] By defined we mean that when Keygen is run on global secret key MSK', $\mathcal{O}'$ is sufficient to answer all of Keygen's queries. Note that c_4 is large enough to allow $\mathcal{O}'$ to contain all the offline values that are needed to satisfy this condition.

[4] We write $[g(\cdot) = \mathrm{pk}] \in L$ to denote that there exists an sk such that $[g(\mathrm{sk}) = \mathrm{pk}] \in L$.

The oracle. The oracle $\mathcal{O}''$ works on a query α as follows:

1. If $\mathcal{O}'(\alpha)$ is defined then reply with $\mathcal{O}'(\alpha)$.

2. Else If α is a query of the form $e(\mathrm{pk}, \cdot)$ (or $d(\mathrm{sk}, \cdot)$) where $\mathrm{pk} \in W \setminus L_g$ (or there exists $\mathrm{pk} \in W \setminus L_g$ such that $[g(\mathrm{sk}) = \mathrm{pk}] \in \mathcal{O}'$) then $\mathcal{O}''$ generates the answer randomly and records it in $\mathcal{O}'$. More precisely, if α is of the form $e''(\mathrm{pk}, x)$ where pk is an internal trapdoor permutation, or of the form $d''(\mathrm{sk}, \alpha)$ where $g'(\mathrm{sk})$ is an internal trapdoor permutation, then $\mathcal{O}''$ emulates a random permutation independent of the actual oracle $\mathcal{O}$. Namely, it responds randomly (and consistently), ensuring that $e''(\mathrm{pk}, \cdot)$ is a permutation and $d''(\mathrm{sk}, \cdot)$ is its inverse, where $g'(\mathrm{sk}) = \mathrm{pk}$. By consistently we mean that any new generated value α (say $e''(\mathrm{pk}, x) = \alpha$) is recorded by adding the entries $e'(\mathrm{pk}, x) = \alpha$ and $d'(\mathrm{sk}, \alpha) = x$ to $\mathcal{O}'$.

3. Else do the following:

   (a) If α is a query of the form $d(\mathrm{sk}, y)$ such that there exists pk such that $[g(\mathrm{sk}) = \mathrm{pk}] \in \mathcal{O}'$ and there exists sk' such that $[g(\mathrm{sk}') = \mathrm{pk}] \in L$ then rewrite α as $d(\mathrm{sk}', y)$. This is to ensure that if we have a correct trapdoor for pk we use it, and not some other incorrect trapdoor that we invented during the offline phase.

   (b) Get the answer from the actual oracle: $\beta \leftarrow \mathcal{O}(\alpha)$, and return β (recall that $[\mathcal{O}(\alpha) = \beta]$ is added to L).

This concludes the description of our adversary. The following lemma directly implies our theorem:

Lemma 2.1.5. *When $2^\lambda > q^{c_5}$ our adversary correctly decrypts the challenge ciphertext with probability at least $\varepsilon - \left(\frac{2}{q^{c_3-c_1-1}} + \frac{3}{eq^{c_2-c_4}} + \frac{2}{q^{c_1-1}} + \frac{10q^{c_1+c_2+c_3+c_4+2}}{2^\lambda}\right)$.*

Proving Lemma 2.1.5 is the main part of the rather technically involved proof. Its details are given in Section 2.1.5

2.1.4 Proof outline for Lemma 2.1.5

In this section we give a high level overview of the proof of Lemma 2.1.5, and highlight the main issues that cause the complex description of our adversary. Below we describe three main problems that our adversary has to deal with in order to successfully simulate the decryption of the challenge ciphertext. Each of the three steps of the adversary deals with one of the problems described below.

Problem 1. The first and somewhat isolated issue is the fact that the challenge ciphertext is computed using the actual oracle while the adversary simulates the decryption using a hybrid between the actual oracle and the small partial oracle $\mathcal{O}'$ which was "invented" offline. Suppose that for some query q which is asked during the computation of the challenge we have $\mathcal{O}'(\mathsf{q}) \neq \mathcal{O}(\mathsf{q})$. Then, this fact may be discovered by the simulated decryption algorithm, and it will refuse to decrypt. This type of inconsistency occurs with small probability due to step 2 "discovering frequent queries" of the adversary. During this step the adversary repeatedly encrypts a random bit to the challenge identity, and records the oracle queries and answers that appear during this computation. Since this is repeated many times the adversary discovers the portions of the oracle that are frequently accessed during such encryptions. The oracle $\mathcal{O}'$ is then chosen in a manner that is consistent with all the information that the adversary has learned about the actual oracle, therefore $\mathcal{O}'$ will be consistent with all the frequent queries and answers that may appear during the computation of the challenge. $\mathcal{O}'$ may contain entries which are not consistent with the actual oracle. However, these queries appear infrequently during encryptions, and so are unlikely to appear during the computation of the challenge ciphertext.

Problem 2. The second issue is that the adversary may be unlucky in its of choice of the challenge identity ID_*. For example, consider a degenerate IBE which securely encrypts messages to one fixed identity and sends the plaintext for every other identity. If the adversary accidentally picks that identity to be challenged on, it will not succeed in decrypting the challenge ciphertext. More generally, our adversary may select an ID_* for which there is a trapdoor that is needed with high probability during decryption, and that trapdoor appears only with low (or zero) probability when encrypting and decrypting ciphertexts for the other identities. In this case, this crucial trapdoor will not be discovered during step 1 ("discovering important trapdoors"), and the adversary will fail to decrypt the challenge ciphertext. However, the adversary will choose an unlucky challenge identity with only a small probability. This is so since if we look at the computation of an encryption and decryption of a random bit for q^{c_1} identities then at most q of these computations may reveal new trapdoors for trapdoor permutations that are embedded in the public parameters. If an identity is chosen at random then with probability $1 - q/q^{c_1}$ its computation will not contain any trapdoors which do not also appear in the computations of the other identities.

Problem 3 (main technical difficulty). The final, and most complex issue is the accuracy of the adversary's simulation of the decryption of the challenge ciphertext. In the previous paragraph we implicitly assumed that unless a new trapdoor is needed, the adversary perfectly simulates the decryption. This assumption is false. However, we will show that the adversary

can do a very good simulation, which is sufficient for our purpose. The decryption of the challenge ciphertext is simulated using the hybrid oracle $\mathcal{O}''$ which is a combination of the partial oracle $\mathcal{O}'$ that was chosen offline, and the actual oracle $\mathcal{O}$. The reason that the adversary's simulation is not perfect is that the partial oracle $\mathcal{O}'$ is chosen to be consistent with the adversary's knowledge of the actual oracle *before* it starts simulating the decryption. New points of the actual oracle that are revealed during the simulation may reveal that the oracle $\mathcal{O}'$ was not chosen from the right distribution, and cause the decryption algorithm to misbehave. This is a very subtle point in the analysis, and deserves more discussion. Consider all the oracle queries that are asked before the adversary chooses its final partial oracle $\mathcal{O}'$. Most of these queries were asked by the adversary, and therefore appear in L. However, some of the queries were asked by the challenger while it computed the private keys for identities $\mathsf{ID} \neq \mathsf{ID}_*$ and the public parameters. Although the private keys and the public parameters are given to the adversary, the queries that were asked while computing them are not. Thus, the adversary obtains some partial information (encoded in the public parameters and private keys) about the queries that were asked.

Without describing a complete example, consider the following simplified scenario: suppose that the public parameters PP contain the answer to one of two queries. That is, PP contains either $\mathcal{O}(\mathfrak{q})$ or $\mathcal{O}(\mathfrak{q}')$, for some $\mathfrak{q} \neq \mathfrak{q}'$, each with probability $1/2$. Now, suppose that in a given run of the experiment PP contains $\mathcal{O}(\mathfrak{q}) = \alpha$ but the adversary guesses that it contains $\mathcal{O}(\mathfrak{q}')$, and sets $\mathcal{O}'(\mathfrak{q}') = \alpha$. This is fine as long as once $\mathcal{O}'$ is fixed, the query $\mathcal{O}(\mathfrak{q})$ is not asked. Otherwise, if $\mathcal{O}(\mathfrak{q})$ is asked then in the adversary's simulated world we get $\mathcal{O}(\mathfrak{q}) = \mathcal{O}(\mathfrak{q}') = \alpha$. This is an unlikely event that can be used by D to abort decryption when run relative to $\mathcal{O}''$. Moreover, the adversary will guess that query input wrongly half the time.

To avoid the situation described above we have the repetitions of step 3. Indeed, once the query $\mathcal{O}(\mathfrak{q})$ is asked, and the mapping $\mathcal{O}(\mathfrak{q}) = \alpha$ is added to L, the adversary is very unlikely to set $\mathcal{O}(\mathfrak{q}') = \alpha$. The total number of queries that are asked by the challenger and not seen by the adversary is at most q^{c_1+1} (q queries for each private key, and for the public parameters), thus if we repeat step 3 sufficiently many times before decrypting the challenge ciphertext we can be assured that the above situation will not arise. Unfortunately, we do not know when one of the hidden queries is discovered. We solve this by repeating step 3 a random number of times, and at the last repetition we decrypt the challenge ciphertext. This can be thought of as repeating step 3 q^{c_3} times, and plugging the challenge ciphertext at a random repetition. There can be at most q^{c_1+1} repetitions during which we discover new hidden queries, and so the probability that we try to decrypt the challenge in one of these repetitions is at most q^{c_1+1}/q^{c_3}.

We have described three problems that may cause our adversary to fail, and our way of dealing with each of the problems. This concludes the intuitive overview of our proof. Formally,

we describe four experiments where the first experiment is the IBE security game played with our adversary, and the last experiment does not involve an adversary, and uses the correct oracle and private key to decrypt the message. We show that the distributions on the transcripts of the experiments are close to each other (in some sense), and thus conclude that the adversary decrypts the challenge ciphertext with probability which is close to the correctness guarantee ε of the IBE.

2.1.5 Proof of Lemma 2.1.5

Probabilistic Lemmas

Fact 2.1.6. *Let $X_1, \ldots, X_{n+1}$ be independent Bernoulli random variables, where $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$ for $i = 1, \ldots, n+1$ and some $p \in [0, 1]$. Let $\mathcal{E}$ be the event that the first n variables are sampled at 1, but X_{n+1} is sampled at 0. Then*

$$\Pr[\mathcal{E}] \leq \frac{1}{e \cdot n}$$

Note that the bound is independent of p.

Proof. Since the variables are independent, event $\mathcal{E}$ happens with probability $p^n(1-p)$. This quantity is maximized at $p = n/(n+1)$. Hence,

$$\Pr[\mathcal{E}] \leq \left(\frac{n}{n+1}\right)^n \cdot \frac{1}{n+1} = \left(\frac{n}{n+1}\right)^{n+1} \cdot \frac{1}{n} \leq \frac{1}{e \cdot n}$$

The last inequality follows since $(n/(n+1))^{n+1} < 1/e$ for all $n \geq 0$. □

Fact 2.1.7. *Let Ω be a probability space, and let f be function with domain Ω. Consider the following experiment: 1. sample x from Ω; 2. sample x' from Ω conditioned on $f(x) = f(x')$. Then, for every $y \in \Omega$, $\Pr[x' = y] = \Pr[x = y]$.*

The following fact captures a property of the random oracle g: after querying a polynomial number of points in g, it is infeasible to output a pair (sk, pk) such that sk is not one the points that were queried and $g(sk) = pk$.

Fact 2.1.8. *Let $\lambda \in \mathbb{Z}^+$, and $g : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a random oracle then for every computationally unbounded algorithm B making queries to g, $\Pr[(sk, pk) \leftarrow B^g(\lambda);\ sk \in \{0,1\}^\lambda \wedge$ sk was not queried by $B \wedge g(sk) = pk] \leq \frac{1}{2^\lambda}$.*

A Series of Experiments

We describe a sequence of four experiments, starting with the security experiment of the IBE such that in each experiment the adversary's probability of success is polynomially close to

its probability of success in the next experiment, and in the last experiment this probability is exactly the same as the correctness guarantee of the IBE. With the exception of the last experiment, for each experiment $i \in \{0, 1, 2\}$ we are interested in the following random variables:

- T^i_{r-init}: the transcript of the challenge step where the private keys for $\mathsf{ID} \neq \mathsf{ID}_*$, and the public parameters PP are computed and given to the adversary. We do not include the computation of the challenge ciphertext C_* in this transcript. The computation of C_* appears in T^i_* which is described below.
- T^i_{ids}: the transcript of step 1 of the adversary where it encrypts and decrypts a single random bit for each $\mathsf{ID} \neq \mathsf{ID}_*$.
- T^i_{freq}: the transcript of step 2 of the adversary where it encrypts a random bit to ID_* q^{c_2} times.
- T^i_{init}: this is the transcript induced by the adversary's choice of MSK' and $\mathcal{O}'$ in the last repetition of step 3 on the computation of $\mathsf{Setup}^{\mathcal{O}'}(\mathsf{MSK}')$ and $\mathsf{Keygen}^{\mathcal{O}'}(\mathsf{MSK}', \mathsf{ID})$ for $\mathsf{ID} \in S$ (including ID_*). In other words, T^i_{init} is the randomly chosen guess of the adversary for the value of T^i_{r-init} (recall that the adversary does not see the transcript T^i_{r-init}).
- T^i_*: this is the transcript of the online phase of the last round of step 3, namely the encryption and decryption of the challenge ciphertext.
- $T^i = (T^i_{r-init}, T^i_{ids}, T^i_{init}, T^i_*)$.

We now proceed to describe the experiments:

Experiment Exp_0. Exp_0 is the security experiment of the IBE, played with our adversary $\mathcal{A}$.

Experiment Exp_1. Exp_1 is the same as Exp_0 except that the challenge ciphertext is computed using the oracle $\mathcal{O}''$, which is defined after the offline phase of the last repetition of step 3. That is, in the last round of step 3, the ciphertext to be decrypted is computed using $\mathcal{O}''$, which is defined using the partial oracle $\mathcal{O}'$ that is chosen in the offline phase.

Experiment Exp_2. Exp_2 is the same as Exp_1 except that the oracle $\mathcal{O}''$ never uses the actual oracle $\mathcal{O}$. Instead, whenever it needs to answer a query for which the answer is not determined by $\mathcal{O}'$ or by the answers to the previous queries, $\mathcal{O}''$ generates an answer randomly from the appropriate distribution. More precisely, to answer a query α in Exp_2, the oracle $\mathcal{O}''$ works as follows:

1. If $\mathcal{O}'(\alpha)$ is defined then reply with $\mathcal{O}'(\alpha)$.

2. If α is of the form $g''(sk)$ then choose a random value $\mathrm{pk} \in \{0,1\}^\lambda$, add the mapping $g'(\mathrm{sk}) = \mathrm{pk}$ to $\mathcal{O}'$ and return pk.
3. If α is of the form $e''(\mathrm{pk}, x)$ (or $d''(\mathrm{sk}, y)$) then $\mathcal{O}''$ chooses a random value $y \in \{0,1\}^\lambda$ ($x \in \{0,1\}^\lambda$) such that $e'(\mathrm{pk}, \cdot)$ ($d'(\mathrm{sk}, \cdot)$) remains a permutation, adds the mapping $e'(\mathrm{pk}, x) = y$ ($d'(\mathrm{sk}, y) = x$) to $\mathcal{O}'$, and returns y.
4. In all of the above cases $\mathcal{O}''$ maintains the following property of $\mathcal{O}'$: for every sk such that $g'(\mathrm{sk}) = \mathrm{pk}$ is defined we require that for every $x \in \{0,1\}^\lambda$, $e'(\mathrm{pk}, x) = \alpha$ is defined if and only if $d'(\mathrm{sk}, \alpha) = x$ is defined. This is done by adding the necessary additional mappings to $\mathcal{O}'$ after answering each query.

Experiment Exp_3. The final experiment does not involve an adversary and uses the correct oracles and private keys:

1. Let $S = \{1, \ldots, q^{c_1}\}$ choose at random $\mathsf{MSK} \in_R \{0,1\}^n$, $\mathsf{ID}_* \in_R [q^{c_1}]$, and compute $\mathrm{PP} \leftarrow G(\mathsf{MSK})$.
2. For every $\mathsf{ID} \in S$ compute $\mathsf{SK}_{\mathsf{ID}} \leftarrow K(\mathsf{MSK}, \mathsf{ID})$.
3. For every $\mathsf{ID} \neq \mathsf{ID}_*$, $\mathsf{ID} \in S$ choose at random $r \in_R \{0,1\}^\lambda$, $b \in_R \{0,1\}$, and compute $C \leftarrow E(\mathrm{PP}, \mathsf{ID}, b; r)$. Then, compute $D(\mathsf{SK}_{\mathsf{ID}}, C)$.
4. For ID_*:
   (a) Choose at random $r \in_R \{0,1\}^\lambda$, $b \in_R \{0,1\}$, and compute $C_* \leftarrow E(\mathrm{PP}, \mathsf{ID}_*, b; r)$.
   (b) Compute and output $b' \leftarrow D(\mathsf{SK}_{\mathsf{ID}_*}, C_*)$

We define the random variable $T^3 = (T^3_{ids}, T^3_{init}, T^3_*)$ as follows: T^3_{ids} is the transcript of step 3; T^3_{init} is the transcript of steps 1 and 2; and T^3_* is the transcript of step 4.

Additional Notation

Throughout the proof we consistently use the following notational conventions.

Transcripts. We will often refer to the transcript of an experiment. A transcript contains all the inputs to the IBE algorithms that were executed during this experiment, as well as the sequence of oracle queries that were asked by the IBE algorithms, and the corresponding answers.

Oracle descriptions. Let $\mathcal{O}' = (g', e', d')$ be a partial trapdoor permutation oracle. We will treat $\mathcal{O}$ intermittently as either a set or a function. We write $[\alpha \mapsto \beta] \in \mathcal{O}'$ if and only if $\mathcal{O}'(\alpha) = \beta$. For a more specific mapping such as $g'(\mathrm{sk}) = \mathrm{pk}$ we write $[g(\mathrm{sk}) = \mathrm{pk}] \in \mathcal{O}'$ if and only if $g'(\mathrm{sk}) = \mathrm{pk}$. We will sometimes use wild cards when describing mappings.

For instance, we write $[g(\sim) = \text{pk}] \in \mathcal{O}'$ if and only if there exists an $\text{sk} \in \{0,1\}^\lambda$ such that $g'(\text{sk}) = \text{pk}$.

Fictional Mappings We denote by $L' \subseteq \mathcal{O}'$ the set of mappings in $\mathcal{O}'$ that the adversary has invented. A mapping is "fictional" if it is not implied by the information that L contains about the actual oracle $\mathcal{O}$. For example: if $[g(\sim) = pk] \notin L$ but $g'(\text{sk}) = \text{pk}$ then since the adversary does not know the correct pre-image of pk under g, the mapping $g'(\text{sk}) = \text{pk}$ is fictional.

Classification of Trapdoor Permutations. Recall the transcripts T_{init} and T_{r-init} that we defined for our experiments. Given a transcript T which is either T^i_{init} for $0 \leq i \leq 3$ or T^i_{r-init} for $0 \leq i \leq 2$ we define the following generalized versions of the random variables U, V, W that were defined in Section 2.1.3:

- **Embedded trapdoor permutations.** T contains the transcript of a computation $\mathsf{Setup}^{\mathcal{O}}(\mathsf{MSK})$ for some (potentially partial) oracle $\mathcal{O}$ and master secret key MSK. Then, we define

$$U(T) = \big\{\text{pk}|\exists \text{sk s.t. } g(\text{sk}) \text{ is asked by } \mathsf{Setup}^{\mathcal{O}}(\mathsf{MSK}) \text{ with answer pk}\big\}$$

- **Internal trapdoor permutations.** T contains the transcript of a computation $K^{\mathcal{O}}(\mathsf{MSK}, \mathsf{ID})$ for $\mathsf{ID} \in S$ for some (potentially partial) oracle $\mathcal{O}$ and master secret key MSK. Then, we define

$$V(T) = \big\{\text{pk}|\exists \text{sk}, \mathsf{ID} \text{ s.t. } g(\text{sk}) \text{ is asked by } K^{\mathcal{O}}(\mathsf{MSK}, \mathsf{ID}) \text{ with answer pk}\big\}$$
$$W(T) = V(T) \setminus U(T)$$

- **Trapdoors discovered during encryption/decryption.** Recall that T^i_{ids} for $0 \leq i \leq 3$ is the transcripts of an encryption and decryption of a single bit to each identity in $S \setminus \{\mathsf{ID}_*\}$, and T^i_* is the transcript of an encryption and decryption of a bit for ID_*. Let T be either T^i_{ids} or T^i_*. We define the set of pk's for which we have discovered trapdoors as follows:

$$Z(T, \mathsf{ID}) = \{\text{pk}| \text{ a query of the form } [g(\sim) = \text{pk}] \text{ is asked during encryption/decryption for } \mathsf{ID}\}$$

Properties of Exp_0

Before comparing the above experiments we will show three properties of Exp_0 that will become essential in the comparison.

First property. Consider the set of trapdoor permutations which are generated by Keygen for some ID but not by Setup. We show that the probability that such a trapdoor is used by the encryption algorithm is small.

Before proving this let us consider the consequences of the negation of this claim. Suppose that there was a way to allow the encryption algorithm to use trapdoor permutations which are generated by Keygen. Then, we could construct a secure IBE by simply assigning one such trapdoor permutation to each identity. The encryption algorithm would use the trapdoor permutation to encrypt and the key generation for this identity would generate the corresponding trapdoor which would allow the holder of that identity's private key to decrypt. Thus, this claim is necessary in a direct way for our theorem.

Claim 2.1.9. *Let $\mathcal{E}_1$ be the event that there is a mapping $pk \in W(T^0_{init}) \setminus L_g$ such that $[e(pk, \sim) = \sim] \in T^0_{freq}$. Then,* $\Pr[\mathcal{E}_1] \leq \frac{q(q^{c_1}+1)}{2^\lambda}$.

Proof. Intuitively, since Keygen can be executed after all encryptions are computed, to make $\mathcal{E}_1$ occur one would have to first select a TP pk and then find its trapdoor. However, since g is a random oracle this can be done with only negligible probability.

Let L_{freq} be the subset of L_g that was constructed during step 2 ("discovering frequent queries"). Let $\mathcal{E}'_1$ be the event that there is a mapping $\mathrm{pk} \in W(T^0_{r-init}) \setminus L_{freq}$ such that $[e(\mathrm{pk}, \sim) = \sim] \in T^0_{freq}$. Let $p' := \Pr[\mathcal{E}'_1]$. We will first prove the bound for the event $\mathcal{E}'_1$, and then use the fact that the adversary chooses MSK' and $\mathcal{O}'$ from the correct distribution to obtain the bound on $\Pr[\mathcal{E}_1]$.

We describe an algorithm B that given access to a random oracle g outputs with probability close to p' a pair $(\mathrm{sk}, \mathrm{pk})$ such that $g(\mathrm{sk}) = \mathrm{pk}$, and the algorithm hasn't queried g on sk. By Fact 2.1.8 this happens with small probability. The algorithm B proceeds as follows. Given oracle access to g it simulates experiment Exp_0 up to (and including) step 2 ("discovering frequent queries") of the adversary. Whenever the oracle g is queried by one of the IBE algorithms, B forwards the query to its own oracle g. Otherwise, B generates all answers to oracle queries from the appropriate distributions.

During the simulation B computes all the encryptions first, before running the key generation algorithm for any of the identities. B then starts simulating the key generation algorithms. At a random query α B stops. B then chooses at random one of the queries that were asked by the encryption algorithm for ID_* during the simulation of step 2 ("discovering frequent queries"). Let β be that query. If α is not of the form $g(\mathrm{sk})$ or if it is not the first time that α is queried then B aborts. If β is not of the form $e(\mathrm{pk}, \cdot)$ B also aborts. Otherwise, B outputs the pair $(\mathrm{sk}, \mathrm{pk})$.

Now, suppose that $\mathcal{E}'_1$ occurred. Then, the probability that B chooses the query $g(\mathrm{sk})$ that

appears in $W(T^0_{r-init}) \setminus L_{freq}$, and the query $e(\mathrm{pk}, \sim)$ that appears in T^0_{freq}, is at least $1/q(q^{c_1} + 1)$. Thus, the total probability that B outputs a valid pair $(\mathrm{sk}, \mathrm{pk})$ is at least $1/q(q^{c_1}+1) \cdot p'$. Therefore, from Fact 2.1.8, we have that $p' \leq \frac{q(q^{c_1}+1)}{2^\lambda}$.

To calculate the probability of $\mathcal{E}_1$ consider the transcripts T^0_{r-init}, T^0_{freq}, and T^0_{init}. Let $t_1 = (T^0_{r-init}, T^0_{freq})$ and $t_2 = (T^0_{init}, T^0_{freq})$. We will show that the pairs t_1 and t_2 are identically distributed.

The pair t_2 can be thought of as the adversary's guess for the value of t_1 after obtaining some partial information about it. More precisely, the adversary sees T^0_{step2} and some additional information about T^0_{r-init} such as the private keys, the public parameters, and the information obtained through step 1 and all but the last round of step 3. Then, in the offline phase of the last round of step 3 the adversary chooses at random $\mathcal{O}'$ and MSK' which are consistent with the previously learnt information about T^0_{r-init} and specify the value of T^0_{init}. Let I be the randomness that specifies the additional information about T^0_{r-init} that is obtained by the adversary. The adversary sees some partial information $\delta = f(T^0_{r-init}, T^0_{freq}, I)$ and chooses at random a tuple $(T^0_{init}, T^0_{freq}, I)$ such that $f(T^0_{r-init}, T^0_{freq}, I) = \delta$. Therefore, by Fact 2.1.7 the distributions on t_1 and t_2 are identical.

The above shows that the probability that there is a mapping $[g(\mathrm{sk}) = \mathrm{pk}] \in W(T^0_{init}) \setminus L_{freq}$ such that $[e(\mathrm{pk}, \sim) = \sim] \in T^0_{freq}$ is equal to p'. In the definition of $\mathcal{E}_1$ we consider only $\mathrm{pk} \in W(T^0_{init}) \setminus L_g$. Since $L_{freq} \subseteq L_g$, we get that $p \leq p'$. □

Second property. Consider all pk's for which $\mathcal{O}'$ contains mappings of the form $[e(\mathrm{pk}, \sim) = \sim]$ but no mapping of the form $[g(\sim) = \mathrm{pk}]$. That is, we have partially defined the permutation $e(\mathrm{pk}, \cdot)$ but have not defined a trapdoor for it. We are now concerned about discovering a trapdoor for pk through the actual oracle $\mathcal{O}$. This is a bad event since $\mathcal{O}'$ contains only mappings of the form $[e(\mathrm{pk}, \sim) = \sim]$ but not their inverses (recall that if $[g(\mathrm{sk}) = \mathrm{pk}] \in \mathcal{O}'$ then $[e(\mathrm{pk}, x) = y] \in \mathcal{O}'$ if and only if $[d(\mathrm{sk}, y) = x] \in \mathcal{O}'$). The following claim shows that the event that a trapdoor is discovered for such a pk happens with small probability.

Claim 2.1.10. *Let $\mathcal{E}^i_2$ be the event that there exists a pk such that $[e(pk, \sim) = \sim] \in \mathcal{O}'$, $pk \notin U(T^i_{init}) \cup W(T^i_{init})$, and there exists sk such that $[g(sk) = pk] \in T^i_*$. Then, for $i \in \{0, 1, 2\}$ $\Pr[\mathcal{E}^i_2] \leq \frac{q^{c_4+c_2}}{2^\lambda}$.*

Proof. The proof for $i \in \{0, 1\}$ is similar to the first half of the proof of Claim 2.1.9 and is omitted.

For $i = 2$, recall that whenever a query $g(\mathrm{sk})$ is made to $\mathcal{O}''$ in Exp_2 a random answer is chosen uniformly from $\{0,1\}^\lambda$. Since there are at most $2q$ queries, the probability that in one of them the answer is a pk as described in the statement of the claim is at most $q^{c_4+2}/2^\lambda$. □

Third property. The third property of Exp_0 that we show captures the reason that we repeat step 3. Recall from our intuitive discussion in the beginning of the analysis that we are concerned about queries for which the adversary has gained some partial but significant amount of information about their answer. The only queries during the experiment that are not seen by the adversary, yet they affect the choices of $\mathcal{O}'$ are the queries that are asked by the challenger during the computation of the public parameters PP and the private keys $\mathsf{SK}_{\mathsf{ID}}$ for $\mathsf{ID} \neq \mathsf{ID}_*$. More precisely, we call a query α hidden if its answer is determined by one of the queries described above:

Definition 2.1.11. A query α is *hidden* if one of the following holds:

1. $\alpha \in T^0_{r-init} \setminus L$.
2. α is of the form $e(\mathrm{pk}, x)$, and there exist sk, y such that $[g(\mathrm{sk}) = \mathrm{pk}], [d(\mathrm{sk}, y) = x] \in T^0_{r-init} \setminus L$ (recall that by our convention when $d(\mathrm{sk}, \cdot)$ is queried, $g(\mathrm{sk})$ is also queried).
3. α is of the form $d(\mathrm{sk}, x)$, and there exists an x such that $[g(\mathrm{sk}) = \mathrm{pk}], [e(\mathrm{pk}, x) = y] \in T^0_{r-init} \setminus L$.

The following claim states that during the last round of step 3 in which the challenge ciphertext is decrypted no hidden queries are asked.

Claim 2.1.12. *Let $\mathcal{E}^i_3$ be the event that a hidden query α appears in T^i_*. Then, $\Pr[\mathcal{E}^0_3] \leq \frac{1}{q^{c_3 - c_1 - 1}}$ where the probability is over all the random choices in Exp_0, including the number of rounds k of step 3.*

Proof. Consider step 4 of the adversary. We choose a random $1 \leq k \leq q^{c_3}$, and repeat a certain process k times, replacing the computation of the ciphertext with the challenge ciphertext in the kth round.

Notice that at each round we compute the ciphertext exactly the same way as the challenge ciphertext is computed. Namely, we use $\mathcal{O}$, and an independently chosen random bit b and random tape r. Thus, the adversary can be thought of as performing q^{c_3} rounds of step 4, plugging the challenge ciphertext at a random round k, and discarding all the information obtained after that round.

In the initialization step the challenger computes $G(\mathsf{MSK})$ and $K(\mathsf{MSK}, \mathsf{ID})$ for $\mathsf{ID} \neq \mathsf{ID}_*$, $\mathsf{ID} \in S$. In total the number of queries made by the challenger is at most $\delta = q^{c_1+1}$ (in other words T^0_{r-init} contains $\leq q^{c_1+1}$ queries). Since L is updated during each round of step 4, there can be at most δ rounds in which new queries from T^0_{r-init} are added to L. So, the probability that the kth round is one of these rounds is at most δ/q^{c_3}. □

A Crucial Property of Exp_3

Experiment Exp_3 does not involve an adversary, and has the property that captures our main intuition regarding why a secure IBE cannot be constructed in this model: if we consider a transcript of Exp_3, we will most likely not see any new trapdoors of embedded trapdoor permutations during step 4 (encryption and decryption for ID_*). By new we refer to trapdoors that have not appeared during step 3 (encryption and decryption for $\mathsf{ID} \neq \mathsf{ID}_*$).

The property of Exp_3 that we need is formalized as follows:

Claim 2.1.13. *Let $\mathcal{E}_4^i$ be the event that $Z(T_*^i, \mathit{ID}_*) \cap U(T_{init}^i) \not\subseteq U(T_{init}^i) \cap \bigcup_{\mathit{ID} \neq \mathit{ID}_*} Z(T_{ids}^i, \mathit{ID})$. Then, $\Pr[\mathcal{E}_4^3] \leq 1/q^{c_1-1}$.*

Proof. The algorithm Setup makes at most q queries and so by definition $|U(T_{init}^i)| \leq q$. Thus, there can be at most q choices of ID_* for which we discover trapdoors for new embedded public keys. Since ID_* is chosen randomly from a set of size q^{c_1} the probability that we hit one of these "bad" identities is at most $1/q^{c_1-1}$. □

Comparing The Experiments

Comparing Exp_1 and Exp_0. In the following claim we show that with high probability it does not matter whether we use $\mathcal{O}$ or $\mathcal{O}''$ to compute the challenge ciphertext.

Claim 2.1.14. *Let $\mathcal{O}$ be any trapdoor permutation oracle, $\lambda, n \in \mathbb{Z}^+$, $\mathit{MSK} \in \{0,1\}^n$, $PP = \mathsf{Setup}^{\mathcal{O}}(\mathit{MSK})$, $\mathit{ID}_* \in \{0,1\}^n$, and $\mathcal{O}''$ the oracle created by the adversary in the last repetition of step 3. Then, $\Pr[\mathsf{Enc}^{\mathcal{O}}(PP, \mathit{ID}, b; r) \neq \mathsf{Enc}^{\mathcal{O}''}(PP, \mathit{ID}, b; r)] \leq \frac{1}{e\, q^{c_2-c_4}}$ where the probability is over the randomness of the adversary $\mathcal{A}$ that is used to construct $\mathcal{O}''$, the bit b, and the random tape r.*

Proof. Consider a specific query α to $\mathcal{O}''$. Let $\mathcal{E}_\alpha$ be the event that query α was not issued during step 2 of the adversary, but was issued during the execution of $\mathsf{Enc}^{\mathcal{O}''}(\mathrm{PP}, \mathsf{ID}_*, b; r)$ with (b, r) from the statement of the claim. Fact 2.1.6 shows that $\Pr[\mathcal{E}_Q]$ is at most $1/(eq^{c_2})$.

Now, if $\mathsf{Enc}^{\mathcal{O}''}(\mathrm{PP}, \mathsf{ID}_*, b; r)$ issues a specific query α in L' then clearly α is not in L which means that $\mathcal{A}$ never observed α during the "discovering frequent queries" phase. Therefore, event $\mathcal{E}_\alpha$ happened which means that the probability of issuing α is at most $1/(eq^{c_2})$. Applying this to all $\leq q^{c_4}$ queries in L' shows that the probability that $\mathsf{Enc}^{\mathcal{O}''}(\mathrm{PP}, \mathsf{ID}^*, b; r)$ issues some query in L' is at most $q^{c_4}/(eq^{c_2}) = 1/(eq^{c_2-c_4})$. Thus, with high probability Enc will only issue queries that are either not in $\mathcal{O}'$ or that $\mathcal{O}$ and $\mathcal{O}'$ agree on.

If $\mathsf{Enc}^{\mathcal{O}''}(\mathrm{PP}, \mathsf{ID}_*, b; r)$ issues a query of the form $e''(\mathrm{pk}, \sim)$ such that $[g(\sim) = \mathrm{pk}] \in W \setminus L$ then $\mathcal{O}''$ would generate the answer randomly. By Claim 2.1.9 a query of the form $[e(\mathrm{pk}, \sim) = \sim]$

appears in L_{freq} with probability at most $\frac{q(q^{c_1}+1)}{2^\lambda}$. If no such query appears in L_{freq}, then as before the probability of such a query being issued is at most $1/(eq^{c_2-c_4})$.

Suppose that a query of the form $d''(\mathrm{sk}, \sim)$ is issued where $[g(\mathrm{sk}) = \sim] \in L'$ (i.e. $g'(\mathrm{sk})$ is defined but we haven't seen $g(\mathrm{sk})$). Then, by our convention the query $g''(\mathrm{sk})$ is also issued. However, we have already shown that the probability of such a query being issued is at most $1/(eq^{c_2-c_4})$. □

The above claim immediately implies the following fact:

Fact 2.1.15. *The statistical distance between T_0 and T_1 is at most $\frac{1}{eq^{c_2-c_4}}$.*

Comparing Exp_1 and Exp_2. The comparison of experiments Exp_1 and Exp_2 is by far the most technically involved part of the proof. For the comparison we need some properties of Exp_2 which will only become apparent after we compare it with the next experiment Exp_3. Therefore, we will delay the comparison of Exp_1 and Exp_2 until after the comparison of Exp_1 with Exp_2.

Comparing Exp_2 and Exp_3. We will compare the distributions induced by each of the experiments on triples $(T^i_{init}, T^i_{ids}, T^i_*)$ where $i \in \{2, 3\}$. Clearly, the marginal distributions (T^3_{init}, T^3_{ids}) and $(T^2_{r-init}, T^2_{ids})$ are identical. The adversary learns some partial information[5] about the pair $\gamma = (T^2_{r-init}, T^2_{ids})$, denoted $f(\gamma)$ and then generates its own guess $\gamma' = (T^2_{init}, T^2_{ids})$ for the value of that pair such that $f(\gamma') = f(\gamma)$. From Fact 2.1.7 we directly obtain that the marginal distributions $(T^2_{r-init}, T^2_{ids})$ and (T^2_{init}, T^2_{ids}) are identical.

To see that $(T^2_{init}, T^2_{ids}, T^2_*)$ and $(T^3_{init}, T^3_{ids}, T^3_*)$ are identical notice that the oracle $\mathcal{O}''$ in Exp_2 is defined to extend $\mathcal{O}'$ (and thus the query-answer pairs from (T^2_{init}, T^2_{ids})) using the correct distributions. Thus we have shown

Fact 2.1.16. *The distributions T^2 and T^3 are identical.*

Comparing Exp_1 and Exp_2 (continued). We are now ready to analyze the relationship between Exp_1 and Exp_2. Exp_2 differs from Exp_1 only in the way the oracle $\mathcal{O}''$ works. Therefore, the marginal distributions (T^1_{init}, T^1_{ids}) and (T^2_{init}, T^2_{ids}) are identical. What remains is to compare the distributions T^1_* and T^2_*.

To compare the transcripts T^1_* and T^2_* we have to consider them in greater detail. The transcripts T^1_* and T^2_* consist of the queries and answers that appeared during the encryption of the challenge ciphertext C_* and then its decryption using the fake private key SK'_* and the oracle $\mathcal{O}''$.

[5] In particular the adversary knows the values of T^2_{ids}

For $j \in \{1,2\}$, and $1 \leq i \leq 2q$, let us denote by α_i^j the random variable which is the ith query in T_*^j, and β_i^j is the answer to that query. When there are less than i queries in T_*^j we set $(\alpha_i^j, b_i^j) = (\perp, \perp)$.

Recall the events $\mathcal{E}_2^i$, $\mathcal{E}_3^i$ and $\mathcal{E}_4^i$. Let $\mathcal{E}^i = \mathcal{E}_2^i \vee \mathcal{E}_3^i \vee \mathcal{E}_4^i$. We define two intermediate experiments Exp_1' and Exp_2' to assist us in the analysis. For every random variable that is defined on Exp_j, $j \in \{1,2\}$, we define the same random variable on Exp_j' with a prime superscript. Thus, the queries and answers in Exp_j' are $(\alpha_i^{j'}, \beta_i^{j'})$.

Exp_1' proceeds as Exp_1 up to the first query α_i for which $\mathcal{E}^1$ occurs. When (and if) $\mathcal{E}^1$ occurs the experiment stops, and the rest of the $(\alpha_j^{1'}, \beta_j^{1'})$ for $j \geq i$ are assigned $(\perp, \perp)$. Similarly, Exp_2' proceeds as Exp_2 up to the first query for which $\mathcal{E}^2$ occurs, after which the experiment stops.

Claim 2.1.17. *The statistical distance between $T^{1'}$ and $T^{2'}$ is at most $\frac{2q^{c_1+c_2+c_3+2}}{2^\lambda}$.*

Proof. We will now compare $\mathsf{Exp}_{1'}$ and $\mathsf{Exp}_{2'}$. Again, all we need to do is compare the sequences $(\alpha_i^{1'}, \beta_i^{1'})_{i\in[2q]}$ and $(\alpha_i^{2'}, \beta_i^{2'})_{i\in[2q]}$. Let $1 \leq i \leq 2q$, and suppose that $\alpha_j^{1'} = \alpha_j^{2'}$ and $\beta_j^{1'} = \beta_j^{2'}$ for all $j < i$. Notice that this implies $\alpha_i^{1'} = \alpha_i^{2'}$ since the ith query is determined by the first $i-1$ queries and answers. We will show that $\beta_i^{1'}$ and $\beta_i^{2'}$ are closely distributed. For convenience we define $\mathcal{O}_i'' = \mathcal{O}' \cup \{\alpha_j \mapsto \beta_j | 1 \leq j < i\}$.

We start by considering the cases where the answer to the ith query $\beta_i^{1'}$ and $\beta_i^{2'}$ is determined by the mappings in $\mathcal{O}_i''$.

1. If $\alpha_i^{1'} = \perp$ then clearly $\beta_i^{1'} = \beta_i^{2'} = \perp$.

2. If there exists a β such that $[\alpha_i^{1'} \mapsto \beta] \in \mathcal{O}_i''$ then $\beta_i^{1'} = \beta_i^{2'} = \beta$.

3. Consider all $\mathrm{pk} \in U(T_{init}^{1'}) \setminus L_g$. Since we are assuming $\neg\mathcal{E}_4^{1'}$ we know that no query of the form $d(\mathrm{sk}, y)$, where $[g(\mathrm{sk}) = \mathrm{pk}] \in \mathcal{O}'$, is asked in $T_*^{1'}$.

4. Consider all $\mathrm{pk} \in W(T_{init}^{1'}) \setminus L_g$. Then, both in $\mathsf{Exp}_{1'}$ and $\mathsf{Exp}_{2'}$ the permutations $e(\mathrm{pk}, \cdot)$ are generated completely offline without using $\mathcal{O}$, and are identically distributed.

5. Consider all $\mathrm{pk} \in (U(T_{init}^{1'}) \cup W(T_{init}^{1'})) \cap L_g$. Then, if $\alpha_i^{1'}$ is of the form $d(\mathrm{sk}, y)$ where $[g(\mathrm{sk}) = \mathrm{pk}] \in \mathcal{O}'$ and there exists a mapping $[e(\mathrm{pk}, x) = y] \in \mathcal{O}_i''$ then either (i) it came from the actual oracle $\mathcal{O}$, or (ii) it came from $\mathcal{O}'$. Since $\mathrm{pk} \in L_g$ we have an sk' such that $[g(\mathrm{sk}') = \mathrm{pk}] \in L$. According to step 3a of the adversary if $\alpha_i^{1'}$ is forwarded to the actual oracle then it is modified to $d(\mathrm{sk}', y)$. Thus, in the first case we get $\beta_i^{1'} = x$ due to the correctness of $\mathcal{O}$, and $\beta_i^{2'} = x$ because x is the correct answer. In the second case we get $\beta_i^{1'} = \beta_i^{2'} = x$ since by the definition of $\mathcal{O}'$

$$[e(\mathrm{pk}, x) = y] \in \mathcal{O}' \iff [d(\mathrm{sk}, y) = x] \in \mathcal{O}'$$

A symmetric argument works when $\alpha_i^{1'}$ is of the form $e(\mathrm{pk}, x)$, with the exception that the query is never modified (only queries of the form $d(\mathrm{sk}, y)$ are modified).

6. Consider all mappings $[g(\mathrm{sk}) = \mathrm{pk}] \in \mathcal{O}_i'' \setminus \mathcal{O}'$. Since we are assuming $\neg\mathcal{E}_2^{1'}$ there are no mappings of the form $[e(\mathrm{pk}, \sim) = \sim] \in \mathcal{O}'$ and since $[g(\mathrm{sk}) = \mathrm{pk}] \notin \mathcal{O}'$ there are no mappings $[d(\mathrm{sk}, \sim) = \sim] \in \mathcal{O}'$. Thus, by the correctness of the oracle $\mathcal{O}$, $\beta_i^{1'}$ gets the correct value and so $\beta_i^{1'} = \beta_i^{2'}$.

Our next step is to consider the cases where the answer to $\alpha_i^{1'}$ is not determined by $\mathcal{O}_i''$.

1. If $\alpha_i^{1'}$ is of the form $g(\mathrm{sk})$. Then, since we are assuming $\neg\mathcal{E}_3^{1'}$ we know $[g(\mathrm{sk}) = \sim] \notin T_{r-init}^{1'}$. This means that in Exp_1' $\alpha_i^{1'}$ is the first time that $g(\mathrm{sk})$ is queried to the actual oracle. Therefore, both $\beta_i^{1'}$ and $\beta_i^{2'}$ are uniformly distributed in $\{0,1\}^\lambda$.

2. Suppose that $\alpha_i^{1'}$ is of the form $e(\mathrm{pk}, x)$. Let $A_{\mathrm{pk}} = \{y | \exists x' \text{ s.t. } [e(\mathrm{pk}, x') = y] \in \mathcal{O}_i''\}$. Let B_{pk} be the set of all strings $y \in \{0,1\}^\lambda$ which were **not** assigned by $\mathcal{O}$ to some x' when $e(\mathrm{pk}, x')$ was queried. Since we are assuming $\neg\mathcal{E}_3^{1'}$, we know $[e(\mathrm{pk}, x) = \sim] \notin T_{r-init}^{1'}$ and that there do not exist sk, y such that $[g(\mathrm{sk}) = \mathrm{pk}], [d(\mathrm{sk}, y) = x] \in T_{r-init}^{1'}$. Therefore, in Exp_1' the answer $\beta_i^{1'}$ is uniformly distributed in B_{pk}.

   In Exp_2', the answer $\beta_i^{2'}$ is uniformly distributed in A_{pk}.

   Notice that $|A_{\mathrm{pk}} \cap B_{\mathrm{pk}}| \geq 2^\lambda - q^{c_1+c_2+c_3+c_4+2}$, $2^\lambda \geq |A_{\mathrm{pk}}|$, and $2^\lambda \geq |B_{\mathrm{pk}}|$. Therefore, we get $|A_{\mathrm{pk}} \triangle B_{\mathrm{pk}|}| \leq q^{c_1+c_2+c_3+c_4+2}$. The statistical distance between $\beta_i^{1'}$ and $\beta_i^{2'}$ is at most $\frac{2|A_{\mathrm{pk}} \triangle B_{\mathrm{pk}}|}{2^\lambda}$.

3. The case where $\alpha_i^{1'}$ is of the form $d(\mathrm{sk}, y)$ is symmetric to the above.

□

We have shown that experiments Exp_1' and Exp_2' proceed almost identically. To conclude the proof we must show that Exp_1 is close to Exp_1' and that Exp_2 is close to Exp_2'.

Claim 2.1.18. *The statistical distance between T^1 and $T^{1'}$ and between T^2 and $T^{2'}$ is at most* $\frac{1}{q^{c_3-c_1-1}} + \frac{1}{eq^{c_2-c_4}} + \frac{1}{q^{c_1-1}} + \frac{4q^{c_1+c_2+c_3+c_4+2}}{2^\lambda}$

Proof. Experiments Exp_1 and Exp_1' proceed identically unless event $\mathcal{E}^1$ occurs in Exp_1 or $\mathcal{E}^{1'}$ occurs in $\mathsf{Exp}_{1'}$. Therefore, event $\mathcal{E}^{1'}$ occurs with the same probability as $\mathcal{E}^1$. Similarly, event $\mathcal{E}^{2'}$ occurs with the same probability as $\mathcal{E}^2$. We have shown:

1. From Claim 2.1.10 we know $\Pr[\mathcal{E}_2^i] \leq \frac{q^{c_4+c_2}}{2^\lambda}$ for $i \in \{0,1,2\}$.

2. From Claim 2.1.12 we know $\Pr[\mathcal{E}_3^0] \leq \frac{1}{q^{c_3-c_1-1}}$.

3. From Claim 2.1.13 we know $\Pr[\mathcal{E}_4^3] \leq \frac{1}{q^{c_1-1}}$.

By applying Fact 2.1.15 we get $\Pr[\mathcal{E}_3^{1'}] \leq \Pr[\mathcal{E}_3^0] + \frac{1}{eq^{c_2-c_4}}$. By applying Fact 2.1.16 we get $\Pr[\mathcal{E}_4^{2'}] = \Pr[\mathcal{E}_4^3]$. Now, by applying Claim 2.1.17 we get

1. $\Pr[\mathcal{E}_3^{2'}] \leq \frac{1}{q^{c_3-c_1-1}} + \frac{1}{eq^{c_2-c_4}} + \frac{2q^{c_1+c_2+c_3+c_4+2}}{2^\lambda}$.
2. $\Pr[\mathcal{E}_4^{1'}] \leq \frac{1}{q^{c_1-1}} + \frac{2q^{c_1+c_2+c_3+c_4+2}}{2^\lambda}$.

Thus we have

$$
\begin{aligned}
\Pr[\mathcal{E}^{1'}] &\leq \Pr[\mathcal{E}_2^{1'}] + \Pr[\mathcal{E}_3^{1'}] + \Pr[\mathcal{E}_4^{1'}] \\
&\leq \frac{q^{c_4+c_2}}{2^\lambda} + \frac{1}{q^{c_3-c_1-1}} + \frac{1}{eq^{c_2-c_4}} + \frac{1}{q^{c_1-1}} + \frac{2q^{c_1+c_2+c_3+c_4+2}}{2^\lambda} \\
&\leq \frac{1}{q^{c_3-c_1-1}} + \frac{1}{eq^{c_2-c_4}} + \frac{1}{q^{c_1-1}} + \frac{4q^{c_1+c_2+c_3+c_4+2}}{2^\lambda} \Pr[\mathcal{E}^{2'}] \qquad \leq \Pr[\mathcal{E}_2^{2'}] + \Pr[\mathcal{E}_3^{2'}] + \Pr[\mathcal{E}_4^{2'}] \\
&\leq \frac{q^{c_4+c_2}}{2^\lambda} + \frac{1}{q^{c_3-c_1-1}} + \frac{1}{eq^{c_2-c_4}} + \frac{2q^{c_1+c_2+c_3+c_4+2}}{2^\lambda} + \frac{1}{q^{c_1-1}} \\
&\leq \frac{1}{q^{c_3-c_1-1}} + \frac{1}{eq^{c_2-c_4}} + \frac{4q^{c_1+c_2+c_3+c_4+2}}{2^\lambda} + \frac{1}{q^{c_1-1}}
\end{aligned}
$$

□

Summing Up. We have described a series of closely related experiment where the first experiment is the security game of the IBE, and the last experiment does not involve an adversary. It is easy to see that in Exp_3 the ciphertext C_* is decrypted with probability ε, which is guaranteed by the correctness property of the IBE. Combining this with Claim 2.1.18, Fact 2.1.16, Fact 2.1.15, and Claim 2.1.17 we obtain our theorem:

Claim 2.1.19. *The probability of the adversary correctly decrypting the challenge ciphertext is at least* $\varepsilon - \left(\frac{2}{q^{c_3-c_1-1}} + \frac{3}{eq^{c_2-c_4}} + \frac{2}{q^{c_1-1}} + \frac{10q^{c_1+c_2+c_3+c_4+2}}{2^\lambda}\right)$

2.2 Towards a black-box separation of IBE from DDH

We ask whether Identity Based Encryption (IBE) can be based in a black-box way on the average-case hardness of Decisional Diffie-Hellman (DDH). Conceptually, this would answer in some sense why many known constructions of IBEs rely on bilinear pairings, which are generalizations of DDH, and not the DDH itself. The DDH assumption is very well-studied, and it finds a surprising number of applications in cryptographic constructions; see e.g. [Bon98] for a not-so-recent survey.

On the technical side, to the best of our knowledge there are no prior black-box separations of any cryptographic primitive from DDH. We present the setting where we aim to do the

black-box separation, and we also present some preliminary results towards this separation. In particular, we show a reduction from an arbitrary IBE system to a restricted IBE. Technically speaking, separating IBE from DDH is significantly more complicated than the separation from TDPs. The reduction we present deals with the main technical difference between this impossibility result and the one described in Section 2.1. Although it is not possible to compare with the details of work in progress, after the description of the model we give a rough idea of what this reduction achieves. There are also several other details and technical differences that need to be taken care of for the full argument to work. We leave those for a future paper.

One interesting feature of this black-box separation is the setting itself. To put things in the proper context let us first recall the DDH assumption.

Assumption 2 (Decisional Diffie Hellman (DDH))**.** There exists an infinite family of encodings of cyclic groups $\{G_n\}$ (say in additive notation), $\{g_n\}, \{p_n\}$, where g_n is a generator of G_n, of prime order p_n. G_n is encoded as a subset of $\{0,1\}^{n^{O(1)}}$, such that (1) there exists a deterministic polynomial time algorithm that given p_n, g_n and $x, y \in G_n$ it computes the value of $x + y$, and (2) for every probabilistic polynomial time adversary A and for sufficiently large n:

$$\left| \Pr_{a,b \in_R \{0,1,\ldots,p_n-1\}}[A(1^n, p_n, g, a{\cdot}g, b{\cdot}g, (ab){\cdot}g) = 1] - \Pr_{a,b,c \in_R \{0,1,\ldots,p_n-1\}}[A(1^n, p_n, g, a{\cdot}g, b{\cdot}g, c{\cdot}g) = 1] \right| \leq \frac{1}{n^k}$$

Our black-box setting uses an oracle which realizes a primitive where DDH holds, and furthermore DDH is naturally related to the primitive in the oracle. At the same time we aim to show that the security of every IBE can be broken in this setting.

The intuition behind the average-case hardness of DDH is that for the family of groups of choice the group elements *look random*. Shoup [Sho97] formalized a model of computation where the group elements *are random*, by an appropriate random embedding of $\mathbb{Z}_n := \mathbb{Z}/n\mathbb{Z}$, $n \in \mathbb{Z}$, into a finite set of strings.

Definition 2.2.1 (Generic Groups Model)**.** Let $n \in \mathbb{Z}^+$ be a parameter, and $2^{n-1} < p < 2^n$ be a prime. Let $\sigma : \mathbb{Z}_p \hookrightarrow S$, $|S| \geq p$, be an injective function inducing a subgroup on the image $\sigma(\mathbb{Z}_p)$. We call σ an *encoding function of* $\mathbb{Z}_p$. A *generic algorithm A for $\mathbb{Z}_p$ on a finite S* is a probabilistic algorithm with access to a random σ and inputs consisting of a string and of group elements of the form $\langle w, \sigma(a_1), \ldots, \sigma(a_k)\rangle$, where $w \in \{0,1\}^*$. A computes as follows: at each step the algorithm specifies a linear combination on the $\sigma(a_i)$'s. This linear combination is determined as a function of w and the group elements $\sigma(x)$ the algorithm A has seen so far (either in the input or in the history of answers to previous queries). The linear combination is a query to the oracle realizing the group operation, which returns the resulting group element

$\sigma(y)$. At the end of the computation A outputs[6] $\langle w', \sigma(b_1), \ldots, \sigma(b_m)\rangle$, $w' \in \{0,1\}^*$.

Note that in this definition A is strongly non-uniform. As usual, the complexity measure is the number of oracle queries. The intuition in this definition is that $\mathbb{Z}_p$ is encoded in a way that no information about other group elements can be extracted by the presented group element (recall that the encoding σ is chosen at random). Intuitively, the only relation that the algorithm may test between group elements is equality. This definition formalizes the intuition that groups for which the Discrete Logarithm (DL) is hard are those satisfying this intuitive property. In fact, Shoup [Sho97] shows that in this model[7] DL is hard. Similarly it can be shown that Assumption 2 holds true in the Generic Groups model. In particular, Public Key Encryption (PKE) exists in this setting.; e.g. the Cramer-Shoup cryptosystem [CS98] is CCA secure given DDH. This is an important distinction from the black-box separation in [IR88]. Let us recall the El-Gamal encryption system, which is based on the hardness of DDH, it is simpler than the Cramer-Shoup, but it is only semantically secure.

Motivating discussion: El-Gamal. We describe a well-known Public-Key Encryption system (conceptually based on the Diffie-Hellman key-exchange). This is known as the El-Gamal encryption system [ElG85]. It is semantically secure assuming that DDH is hard, and thus good enough (see Section 2.1.1 for the security notion we use for IBE) for this motivating discussion. Recall, that a PKE system is a special case of an IBE. It consists of three algorithms: key-generation, encryption, and decryption. Keygen outputs a public PUB and a private key SK. As usual, PUB is used for encryption, and SK is used for decryption. Here is the description of the El-Gamal PKE system.

We describe this PKE system in the Generic Groups Model. Suppose that the three algorithms have access to the same oracle $\mathcal{O}$ for the operations of the group G. For notational simplicity we omit denoting the oracle when no confusion arises; e.g. we write Keygen instead of $\mathsf{Keygen}^{\mathcal{O}}$. We encrypt/decrypt group elements (i.e. the message is just a group element). In the following description, the strings corresponding to randomness are identified by integers.

$\mathsf{Keygen}(g_1, r)$*:* Use the randomness r and the generator g_1 of G to sample an element $h := r \cdot g_1$. Output: PUB $= (h, g_1)$ and $\mathsf{SK} = r$.

$\mathsf{Enc}(\underbrace{(h, g_1)}_{PUB}, g, r')$*:* The group element g is the message we wish to encrypt. Output the cipher $c = (c_1, c_2)$, where $c_1 = r' \cdot g_1$ and $c_2 = r' \cdot h + g$.

[6] This notation does not imply that A knows the b_i's (in fact, in general this is impossible). An element $\sigma(b_i)$ is presented to A as an answer to a query regarding a linear combination of the $\sigma(a_i)$'s.

[7] The model considered by Shoup is equivalent, but notationally less convenient for our purposes.

$\mathsf{Dec}(\underbrace{r}_{SK}, (c_1, c_2))$*:* Output $c_2 - (r \cdot c_1)$.

Correctness is immediate by definition. Keygen outputs public key $(r \cdot g_1, g_1)$; then for a message g, Enc outputs $c = (r' \cdot g_1, r' \cdot (r \cdot g_1) + g)$; and on input c, Dec outputs $r' \cdot (r \cdot g_1) + g - r \cdot (r' \cdot g_1) = g$.

It is also semantically secure in the Generic Groups Model. Its security follows by the hardness of DDH, and DDH is hard in the Generic Groups Model.

Imagine that we modify the above scheme as follows. Partition r into n^k many contiguous blocks, each of which is of polynomial size, and generate the pair:

$$\mathsf{MPUB} = (r_1 \cdot g_1, r_1 \cdot g_2, \ldots, r_{n^k} \cdot g_{n^k}), \quad \mathsf{MSK} = (r_1, r_2, \ldots, r_{n^k})$$

Associate each "subpair" $(r_i g_1, r_i)$ with a particular identity. Then, it is obvious how to extend the El-Gamal cryptosystem to an IBE-like one, where we rename Keygen to Setup, and we construct a new Keygen which issues the "subpairs" for the relevant identities.

By the properties of El-Gamal, it is obvious that this IBE-like system works only for at most n^k identities, and it works securely. However, as the number of identities becomes enough bigger than n^k then no matter how you modify this encryption system, it must be the case that there will be "relations" among the private (secret) keys associated with identities. In our separation argument (which is work in progress) we identify these "relations" as linear relations over appropriate vector spaces, and we construct an adversary that finds them in time polynomial in the maximum number of queries an IBE algorithm makes.

2.2.1 Definitions and notational conventions (differences with Section 2.1).

All algorithms in an IBE are generic with access to the same oracle $\mathcal{O}$. We describe the oracle and relevant notation below. The four algorithms are $(\mathsf{Setup}^{\mathcal{O}}, \mathsf{Keygen}^{\mathcal{O}}, \mathsf{Enc}^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}})$; we omit $\mathcal{O}$ when it is clear from the context. The input to each algorithm is a string and a tuple of group elements: $\langle w, \sigma(a_1), \ldots, \sigma(a_m)\rangle$, and the same holds for the output. As in Section 2.1.1, here too we consider the possibility of WSS-IBEs, with noticeable correctness; where noticeable probability means that there is a constant $c > 0$ such that for sufficiently large n, the probability is $\geq \frac{1}{n^c}$.

In Section 2.1 (and other related works) many parameters are used, e.g. λ is the security parameter, n corresponds to input length etc. In this section we reduce notational clutter and we focus on the useful and typical case[8].

Parameters. Instead of $\lambda, n, q, \varepsilon$ we have:

[8]Recall that this reduction would be part of a general impossibility result, where it makes sense to consider different cases for the proof as a whole.

- n: the security parameter. According to Definition 2.2.1 the group is of order $p \approx 2^n$.
- ε: the IBE is correct with probability $\geq \varepsilon > \frac{1}{2}$ over the randomness in the security experiment.
- k: every algorithm in the IBE makes $\leq n^k$ queries. Let $m := n^k$.

The oracle $\mathcal{O}$. We introduce the notation regarding oracle access; i.e. queries and how to answer them. Let n, p be as in Definition 2.2.1; i.e. $p \approx 2^n$. Let $\sigma : \mathbb{Z}_p \hookrightarrow \{0,1\}^n$ a random embedding. We denote by $G = \sigma(\mathbb{Z}_p)$ the induced group in additive notation. We write $\mathbf{0}$ to denote $\sigma(0)$, the n-bit string which is the identity element of G. Recall that for any finite set S the set of formal sums $\mathbb{Z}_p[S]$ is an $|S|$-dimensional vector space in the usual sense. In particular, we will treat $\mathbb{Z}_p[G]$ as a vector space. Occasionally, we write *linear combinations of elements of G* to refer to formal sums in $\mathbb{Z}_p[G]$. For $\mathbb{Z}_p[G]$ the group structure of G becomes relevant through the *valuation map*. We denote by $\mathcal{O}$ the valuation map $\mathcal{O} : \mathbb{Z}_p[G] \to G$ defined as follows. For $\alpha \in \mathbb{Z}_p$ and $u \in G$, $\alpha g := \underbrace{g +_G g +_G \ldots +_G g}_{\alpha \text{ times}}$, where $+_G$ is the group operation. The formal sum $\alpha_1 g_1 + \alpha_2 g_2 + \ldots + \alpha_k g_k$ valuates through $\mathcal{O}$ to a group element where instead of $+$ we consider $+_G$. Henceforth, we use $+$ instead of $+_G$ when no confusion arises.

In an IBE system the four algorithms have access to the same valuation $\mathcal{O}$, but each algorithm is restricted to different oracle-access in the following sense. If the input to an algorithm $\mathcal{A}$ is $\langle w, g_1, g_2, \ldots, g_m \rangle$, $w \in \{0,1\}^*$ and $g_i \in G$ then every query of $\mathcal{A}$ is of the form $\sum_{i=1}^m \alpha_i g_i$; i.e. the access to oracle $\mathcal{O}$ is restricted to valuations of formal sums of $S = \{g_1, \ldots, g_m\}$. Note that $\mathbb{Z}_p[S]$ is a subspace of $\mathbb{Z}_p[G]$. We also make the convention that the first group element in the input and output (if the output contains any group elements) of a generic algorithm is a generator of the group (i.e. $\neq \mathbf{0}$, since $G \approx \mathbb{Z}_p$).

We use the following terms interchangeably. A *query* q of an algorithm is a formal sum of its input elements. An *answer to a query* q is a valuation.

2.2.2 The Theorem

Let $\mathcal{IBE}^{\mathcal{O}} = (\mathsf{Setup}^{\mathcal{O}}, \mathsf{Keygen}^{\mathcal{O}}, \mathsf{Enc}^{\mathcal{O}}, \mathsf{Dec}^{\mathcal{O}})$ be an Identity Based Encryption system where the algorithms are generic with oracle access to $\mathcal{O}$. When the oracle is clear from the context we omit $\mathcal{O}$ and we write $\mathcal{IBE}$. Recall that $\mathcal{IBE}$ comes with parameters k, ε, such that, for every n each algorithm makes at most n^k queries to $\mathcal{O}$, and $\mathcal{IBE}$ is correct with probability $\geq \varepsilon$.

Restricted Identity Based Encryption. We say that an Identity Based Encryption system $\mathcal{IBE}' = (\mathsf{Setup}', \mathsf{Keygen}', \mathsf{Enc}', \mathsf{Dec}')$ is *restricted* if the Key-generation algorithm, Keygen', outputs secret keys $\mathsf{SK}_{\mathsf{ID}}$ that do not contain any group elements.

Theorem 2.2.2. *Suppose that* $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$ *is a WSS-IBE, correct with probability* $\geq \varepsilon$*. Then, there exists* $\varepsilon' > \frac{1}{2}$ *and a* restricted IBE*,* $\mathcal{IBE}' = (\mathsf{Setup}', \mathsf{Keygen}', \mathsf{Enc}', \mathsf{Dec}')$*, which is a WSS-IBE, correct with probability* $\geq \varepsilon'$*.*

Discussion on the main theorem

Why our main theorem is of any use to the black-box separation? This reduction makes the template of the separation from DDH to look the same as for TDPs. The rest of the result is work in progress and we aren't going to discuss it further. However, there is an independent, explicit, technical reason why we care about this theorem. Theorem 2.2.2 "removes" the group elements from the output of Keygen, which are the secret keys. Now, imagine[9] that we were able to remove the group elements from the output of each of the remaining three algorithms. Then, we claim that by [IR88] we can conclude the impossibility.

The fact that the output of the algorithms contain group elements, is where the main technical complication happens. In particular, if no group elements were to be included in the global public key MPUB, the secret keys $\mathsf{SK}_{\mathsf{ID}}$'s, and the ciphertexts, then the impossibility follows by [IR88], since the group oracle acts just like a random oracle.

Elementary properties used in the reduction

The two ways[10] of considering the group (as formal sums of elements, and as valuations of formal sums) can be "mixed", since $\mathcal{O}$ is in particular a group homomorphism. Here are two trivial consequences (we mention them to emphasize their use).

Fact 2.2.3.

- *Let two queries* $\mathfrak{q}_1 := \sum_{i=1}^{k} \alpha_i u_i + \sum_{i=1}^{l} \beta_i v_i$ *and* $\mathfrak{q}_2 := \sum_{i=1}^{k} \alpha_i u_i + \sum_{i=1}^{l'} \gamma_i h_i$*, where* $\alpha_i, \beta_i, \gamma_i \in \mathbb{Z}_p, u_i, v_i, h_i \in G$*. Then,* $\mathcal{O}(\mathfrak{q}_1) = \mathcal{O}(\mathfrak{q}_2) \iff \mathcal{O}(\sum_{i=1}^{l} \beta_i v_i) = \mathcal{O}(\sum_{i=1}^{l'} \gamma_i h_i)$*.*
- *Let* $\mathfrak{q}_{\mathsf{id}} \in \ker(\mathcal{O})$ *and* $\mathfrak{q}$ *a query. Then,* $\mathcal{O}(\mathfrak{q}) = \mathcal{O}(\mathfrak{q} + \mathfrak{q}_{\mathsf{id}})$*.*

Let us give a preview on some part of the reduction. At a high level $\mathcal{IBE}'$ simulates $\mathcal{IBE}$. The main issue is that in $\mathcal{IBE}'$ we wish to somehow use $\mathcal{IBE}$ so as to answer in Dec' queries involving group elements that appear in $\mathsf{SK}_{\mathsf{ID}}$. But, by definition of a restricted IBE we do not have group elements in the secret keys. There are cases where for the same typical answer there are many corresponding queries, expressed as distinct formal sums. The main technical part

[9] Note that our actual proof in progress does not work this way. It seems technically simpler to directly construct an adversary against *restricted IBEs*.

[10] Loosely speaking, we reduce a general $\mathcal{IBE}$ to $\mathcal{IBE}'$, where $\mathcal{IBE}'$ tries to simulate answers to queries involving group elements it does not know, by acting in part "syntactically" similar to a proof system, and in part it relies on the "semantics" which are the valuations of queries.

of our argument formalizes the following intuition: no matter which these formal sums are, as long as they are"known" by the algorithms in $\mathcal{IBE}$, then they can be obtained by *syntactic* manipulations using Fact 2.2.3. Roughly speaking, our task is to find a sufficient subspace of $\mathsf{ker}(\mathcal{O})$, and sufficiently many ways to express the same frequent answer.

2.2.3 The reduction

Fix an arbitrary $\mathcal{IBE}$. Here is how we define an $\mathcal{IBE}'$ without group elements in the output of Keygen' with the following property. $\mathcal{IBE}'$ makes polynomially more queries than $\mathcal{IBE}$, and it preserves security and correctness within polynomial factors. Recall that $m = n^k$.

Notation for group elements in $\mathcal{IBE}$

We introduce some notation for the group elements in the inputs/outputs of the algorithms for the $\mathcal{IBE}$. We denote by $m_1, m_2, m_3, m_4 \leq m$ the number of group elements in the outputs of $\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec}$.

- Setup: without loss of generality, the private key is the randomness for Setup, and we denote group elements of the public key by: $g_1, \ldots, g_{m_1}$.
- Keygen: we denote the group elements in $\mathsf{SK}_{\mathsf{ID}}$ by: $z_1, \ldots, z_{m_2}$.
- Enc: we denote the group elements of the ciphertext by: $t_1, \ldots, t_{m_3}$.
- Dec: this algorithm does not output group elements; it decrypts the bit encrypted by Enc.

Let the (general) $\mathcal{IBE}$ be the tuple of algorithms $\mathcal{IBE} := (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$. We define $\mathcal{IBE}' := (\mathsf{Setup}', \mathsf{Keygen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- Setup': The same as Setup.
- $\mathsf{Keygen}'(\mathrm{PK}, \mathsf{ID})$: Run $(w, z_1, \ldots, z_m) \leftarrow \mathsf{Keygen}(\mathrm{PK}, \mathsf{ID})$.
  Iteratively run Enc and Dec on ID for a randomly chosen number of iterations from $\{\alpha, \alpha+1, \ldots, \alpha+\beta\}$, where $\alpha = n^{7k+2}, \beta = 2n^{4k}$, for randomly chosen bits to encrypt, and record the $(\mathsf{query}, \mathsf{answer})$ pairs in a table T; we also use the notation $\mathsf{query} \mapsto \mathsf{answer}$.
  Output: $\mathsf{SK}'_{\mathsf{ID}} := (w, T, \bar{z})$, where $\bar{z}$ is the group elements $z_1, \ldots, z_m$ listed as strings.

  Remark: The entries of T are of the form $\sum_{i=1}^{m} \alpha_i z_i + \sum_{i=1}^{m} \beta_i g_i \mapsto s$, where $s \in \{0,1\}^n$ is the $\mathcal{O}$-valuation of the formal sum. We can write the formal sum in terms of the z_i's and g_i's because we are simulating both Enc and Dec; thus, we know to which linear combination of the g_i's each t_i corresponds to. The following remark regards Dec' below.

Because of step (3) when answering oracle queries in Dec', T may be updated to contain entries of the form $\sum_{i=1}^{m} \alpha_i z_i + \sum_{i=1}^{m} \beta_i t_i \mapsto s$, where now the t_i's are the specific t_i's provided in the input of Dec'.

- Enc': The same as Enc.
- $\mathsf{Dec}'(\mathsf{ID}, c, \mathsf{SK}'_{\mathsf{ID}}, g_1, g_2, \ldots, g_m, r)$: The string r is extra randomness. Run $\mathsf{Dec}(\mathsf{ID}, c, \mathsf{SK}_{\mathsf{ID}})$, where $\mathsf{SK}_{\mathsf{ID}} = (w, z_1, \ldots, z_m)$; where the z_i's are strings (not group elements). Here is how to perform this simulation. The main issue is how to answer oracle queries that involve the z_i's group elements; note that $\mathsf{SK}'_{\mathsf{ID}}$ does not contain any group elements - it only contains the z_i's as strings. Ideally, we would like either (i) to find whether the query was "asked" before, or (ii) to rewrite the linear combination of the "z_i's part of the query" as a linear combination of the g_i's (to which Dec' has oracle access). Use T to derive R_T (or simply R) of all equality relations derivable from T as follows. For every two pairs $\mathfrak{q} \mapsto u$ and $\mathfrak{q}' \mapsto u$ in T, we derive the equality relation $\mathfrak{q} - \mathfrak{q}' = \mathbf{0}$. The formal sum $\mathfrak{q} - \mathfrak{q}'$ is called *equality vector*. Note that the formal sums in R are a subset of $\ker(\mathcal{O})$, and they span a sub-vector space of the kernel of $\mathcal{O}$.

Answering oracle queries of Dec': Here is how to valuate an oracle query

$$\mathfrak{q} = \mathfrak{q}^z + \mathfrak{q}^t = \underbrace{\sum_{i=1}^{m} \gamma_i z_i}_{\mathfrak{q}^z} + \underbrace{\sum_{i=1}^{m} \delta_i t_i}_{\mathfrak{q}^t}$$

1. Try to use R to rewrite $\mathfrak{q}^z = \sum_{i=1}^{m} \alpha_i g_i$. If this is possible, let $\mathfrak{q}'$ be the rewritten $\mathfrak{q}$, and answer $\mathcal{O}(\mathfrak{q}')$.
2. If (1) fails then check whether $\mathfrak{q}$ was "asked before" as follows: Consider the equality vectors $E_1, E_2, \ldots, E_M$ (i.e. the entries of R) and denote by E_i^z the part of E_i indexed by the z_i's. Then, iterate through every query q_{old} in T and do the following: check whether there is a linear combination expressing

$$\mathfrak{q}^z = \mathfrak{q}^z_{old} + \sum_{i=1}^{M} \kappa_i E_i^z$$

If there is such a combination[11] check whether $\mathcal{O}(\mathfrak{q}^t) = \mathcal{O}(\mathfrak{q}_{old}^{g \text{ or } t} + \sum_{i=1}^{M} \kappa_i E_i^g)$. And if true then resolve $\mathfrak{q}$ as $T(\mathfrak{q}_{old}) = s$, i.e. $\mathfrak{q}_{old} \mapsto s$ is recorded in T; otherwise, step (2) fails. The notation $\mathfrak{q}_{old}^{g \text{ or } t}$ refers to the fact that $\mathfrak{q}_{old}$ may contain a combination of the g_i's or the t_i's (see step (3) below).

[11] Note that by Fact 2.2.3 any such combination is as good as any other. In particular, it suffices to find and check only one.

3. If (2) also fails, then choose a random string $s \in_R \{0,1\}^n$ and update T by adding $\mathsf{q} \mapsto s$.

Notation for the table: We denote by T_0 the content of the table T at the beginning of the execution of Dec'. A "round" in the execution of Dec' is a simulation of answering a single query made during the simulation of Dec. We denote by T_i the content (state) of T at the end of round i.

Intuitive remarks when answering oracle queries in Dec'. Intuitively, step (1) deals with queries made both by Enc and Dec. It is common practice in encryption to algebraically express at least part of the ciphertext in two different ways. That is, by making such queries, during the simulation of Keygen' the useful relations among the z_i's and the g_i's are revealed. On the other hand step (2) can be thought to deal with queries Dec makes; i.e. formally different queries corresponding to the same group element during the same execution of Dec. The lemmas in Section 2.2.6 clarify the role of these two steps. The last step (3) intuitively asserts that if no algebraic relations can be deduced from T (deduced in the way defined in Dec', then this query should be distributed uniformly; i.e. almost as in σ.

2.2.4 High-level overview of the proof

We show that if $\mathcal{IBE}$ is secure and correct then $\mathcal{IBE}'$ is secure and correct. Security is straightforward. To show that correctness is preserved (with a small loss) we will show two things. First, we show that the way we answer oracle queries in Dec' is consistent to a group oracle(*). Second, we show that answers given by this group oracle are appropriately distributed (**). Note that (**) is a probabilistic statement, whereas (*) is not. We also remark that in order to show (**) we will use (*).

One can easily construct examples of executions where (*) and (**) fail. To that end we identify, an obstruction which corresponds to an event (of high probability) in the experiment, such that conditioned on that event (*) and (**) hold true. Since, (*) is in principle not a probabilistic statement, by "conditioning on an event" we mean that we choose the parameters of the execution restricted to this event.

2.2.5 Proof of Main Theorem

We prove Theorem 2.2.2.

Security is immediate by construction of $\mathcal{IBE}'$.

Fact 2.2.4. *If $\mathcal{IBE}$ is secure then $\mathcal{IBE}'$ is secure. That is, if there exists adversary Adv' which*

breaks $\mathcal{IBE}'$ by making $n^{O(1)}$ many queries, then there exists Adv which breaks $\mathcal{IBE}$ with $n^{O(1)}$ many queries.

For the non-trivial part of Theorem 2.2.2 we show in Theorem 2.2.5 that there is at most a small constant loss in the correctness of $\mathcal{IBE}'$ compared to the correctness of $\mathcal{IBE}$.

Theorem 2.2.5. *Let $\mathcal{IBE}$ and $\mathcal{IBE}'$ be as above, where $\mathcal{IBE}$ is correct with probability $\varepsilon > 1/2$. Then, $\mathcal{IBE}'$ is correct with probability ε', where $\frac{1}{2} < \varepsilon' < \varepsilon$.*

To prove this theorem it is sufficient to show that

(i) the way we answer queries in $\mathcal{IBE}'$ is consistent to a group oracle and

(ii) this oracle, henceforth denoted as $\mathcal{O}'$, is chosen from a distribution sufficiently close to the distribution $\mathcal{O}$ is chosen in $\mathcal{IBE}$.

Presentation difference from Section 2.1

In Section 2.1 the argument is using the informal notion of an "experiment", and it proceeds by considering a sequence of related experiments. Formally speaking, these "experiments" correspond to probability spaces. The argument in Section 2.1 is rigorous, in the sense that it translates in the formal context of probability.

In this section we choose to directly refer to the actual probability spaces. One reason for this is that the correspondence between the events we wish to compare is more precise; i.e. a bijection between events of the same measure is possible, and thus more easily understood in the language of probability spaces. Occasionally, we use the terms "experiment", "transcript" and so on for intuition.

More notation and terminology

Let $\mathsf{Exp}_{\mathcal{IBE}}$ be the security experiment (probability space) associated with $\mathcal{IBE}$, and $\mathsf{Exp}_{\mathcal{IBE}'}$ be the security experiment for $\mathcal{IBE}'$. $\mathsf{Exp}_{\mathcal{IBE}}$ (and $\mathsf{Exp}_{\mathcal{IBE}'}$) is a security experiment in the informal sense of a probabilistic experiment. The same notation $\mathsf{Exp}_{\mathcal{IBE}}$ will be used to refer to the corresponding probability space. Each element of this space has the same probability (i.e. the space is uniform) and it consists of an oracle $\mathcal{O}$, and the necessary amount of randomness for choosing the parameters of the experiment, and running the probabilistic algorithms (realized as deterministic algorithms with randomness in the input). In what follows, instead of considering an oracle contained in each element of the space, we identify this oracle as a (polynomially small) sequence of strings from $\{0,1\}^n$.

We do not directly compare the security experiments for $\mathcal{IBE}$ and $\mathcal{IBE}'$. Instead, we introduce some intermediate experiments for technical convenience. We identify the randomly

chosen oracle $\mathcal{O}$ for the experiment as a finite sequence $\langle u_1, u_2, \ldots, u_{4n^k}\rangle$ of distinct answers to new queries defined as follows.

Definition 2.2.6 (New query)**.** Consider the security experiment for $\mathcal{IBE} = (\mathsf{Setup}, \mathsf{Keygen}, \mathsf{Enc}, \mathsf{Dec})$. Then, its query q has a well-defined discrete logarithm, with respect to the generator given initially to Setup. In the sequence of queries in this execution a *query* $\mathsf{q} \in \mathbb{Z}_p[G]$ *is new* if it has a different discrete logarithm than the previous queries in the sequence. Similarly, we define new queries for the experiment of $\mathcal{IBE}'$.

Since there are four algorithms involved in $\mathsf{Exp}_{\mathcal{IBE}}$ it is sufficient for this finite sequence to be of length $4n^k$. We modify the experiment of $\mathcal{IBE}'$ as follows. Let $\mathsf{Exp}^1_{\mathcal{IBE}'} := \mathsf{Exp}_{\mathcal{IBE}'}$, and define $\mathsf{Exp}^2_{\mathcal{IBE}'}$ the same as $\mathsf{Exp}^1_{\mathcal{IBE}'}$ where the random string r given in Dec' is not a uniformly random string, but rather $r = \langle r_1, r_2, \ldots, r_{n^{O(1)}}\rangle$ where $r_i \in \{0,1\}^n$, and each of the r_i's is distinct from each other and from the oracle answers to new queries u_j's.

Fact 2.2.7. *If $\mathcal{IBE}'$ is correct in $\mathsf{Exp}^2_{\mathcal{IBE}'}$ with probability $\geq \hat{\varepsilon} > 1/2$, then it is correct in $\mathsf{Exp}^1_{\mathcal{IBE}'}$ with probability $\geq \varepsilon$, for some constant $\frac{1}{2} < \varepsilon < \hat{\varepsilon}$.*

Proof. Let $\mathcal{E}_i$ be the following event: r_i is not equal to $r_{i-1}, \ldots, r_1,\ u_1, \ldots, u_{4n^k}$. Then the following identity follows by definition of conditional probability. Let s a fixed, and $|r| = |s|$; we denote by $\Pr_r[r = s]$ as $\Pr[s]$.

$$\Pr_{\mathsf{Exp}^2_{\mathcal{IBE}'}}[s] = \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}}}[s|\mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_{4n^k}] \implies \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[s] \geq \Pr_{\mathsf{Exp}^2_{\mathcal{IBE}'}}[s](1 - \sum_i \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[\bar{\mathcal{E}}_i])$$

In what follows we denote by "s:distinct elements", the fact that each substring s_i corresponding to r_i is distinct and distinct from every u_j.

$$\begin{aligned}
\Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[\mathcal{IBE}' \text{ is correct}] &= \sum_s \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[\mathcal{IBE}' \text{ is correct}|s] \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[s] \\
&\geq \sum_s \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[\mathcal{IBE}' \text{ is correct}|s] \Pr_{\mathsf{Exp}^2_{\mathcal{IBE}'}}[s](1 - \sum_i \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[\bar{\mathcal{E}}_1]) \\
&\geq \sum_{s:\text{distinct elms}} \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[\mathcal{IBE}' \text{ is correct}|s] \Pr_{\mathsf{Exp}^2_{\mathcal{IBE}'}}[s](1 - \sum_i \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[\bar{\mathcal{E}}_1]) \\
&= (1 - \sum_i \Pr_{\mathsf{Exp}^1_{\mathcal{IBE}'}}[\bar{\mathcal{E}}_i]) \sum_{s:\text{distinct elms}} \Pr_{\mathsf{Exp}^2_{\mathcal{IBE}'}}[\mathcal{IBE}' \text{ is correct}|s] \Pr_{\mathsf{Exp}^2_{\mathcal{IBE}'}}[s] \\
&\geq (1 - \frac{n^{O(1)}}{2^n}) \Pr_{\mathsf{Exp}^2_{\mathcal{IBE}'}}[\mathcal{IBE}' \text{ is correct}] \geq \varepsilon
\end{aligned}$$

for some ε, and sufficiently large n. □

Hence, it is sufficient to show that given that $\mathcal{IBE}$ is correct then $\mathcal{IBE}'$ is correct when in the randomness r, every r_i is distinct from every other r_j and u_j.

There is a little more house-keeping work left to do. We wish to make $\mathsf{Exp}^2_{\mathcal{IBE}'}$ and $\mathsf{Exp}_{\mathcal{IBE}}$ identical as probability spaces. We make Keygen in $\mathcal{IBE}$ to query as in Keygen' the pair of algorithms $\mathsf{Enc\text{-}Dec}$, but do nothing with the answers. We also add some extra randomness in Dec, the same amount of randomness that Dec' is using when making up answers to queries in step (3). Let this new experiment be Exp. With these modifications we have more queries, but this can be fixed by adding dummy queries to the algorithms, and replacing k with a bigger constant (which for simplicity we will continue to denote by k).

All of the above, are minor technical points that reduce the problem of comparing $\mathsf{Exp}_{\mathcal{IBE}}$ to $\mathsf{Exp}_{\mathcal{IBE}'}$, to comparing Exp to $\mathsf{Exp}' := \mathsf{Exp}^2_{\mathcal{IBE}'}$. Note that when Exp and Exp' are used to denote probability spaces then they are equal. However, in the informal notion of "experiment" Exp is associated with $\mathcal{IBE}$ and Exp' with $\mathcal{IBE}'$ (this also emphasizes that with the same randomness we consider executions of different algorithms in $\mathcal{IBE}$ and in $\mathcal{IBE}'$).

In the remainder of this section we give the main part of the analysis of correctness.

An obstruction to correct simulation of Dec by Dec'

We define a bad event BAD. Conditioned on $\overline{\text{BAD}}$, the oracle $\mathcal{O}'$ we implement in Dec' corresponds (i.e. it can be extended) to a group oracle, and furthermore it is properly distributed.

Definition 2.2.8. Fix an oracle $\mathcal{O}$. Consider an execution of the security experiment for $\mathcal{IBE}$, i.e Exp. An *answer $u \in G$ is frequent for $\mathcal{O}$* in the execution of $\mathsf{Enc\text{-}Dec}$ if it appears with probability $\geq \frac{1}{n^{7k}}$. An answer which is not frequent is called *infrequent*.

Let $\text{BAD} := \text{BAD}_1 \vee \text{BAD}_2 \vee \text{BAD}_3$ defined for Exp' as follows.

BAD_1: There is an infrequent answer given both in Dec' (excluding the queries made due to step (2)) and in one of the executions of the following: in $\mathsf{Setup}', \mathsf{Enc}'$ or in the simulation of Keygen by Keygen'.

BAD_2: There is a frequent answer not recorded in T during the iterative ($\mathsf{Enc\text{-}Dec}$) phase of Keygen'.

BAD_3: Consider the execution of $\mathsf{Enc}'\text{-}\mathsf{Dec}'$, and let T' be the collection of query-answer pairs. Let $T := T_M$, where M is the last round of Dec'. The event is: $T \cup T'$ induces an equality vector linearly independent from R.

In Section 2.2.7 we show the following lemma.

Lemma 2.2.9. $\Pr[\text{BAD}] \leq f(n)$, *where* $f : \mathbb{Z}^+ \to [0,1]$ *is a function that monotonically decreases to* 0.

In Section 2.2.6 we show that the way we answer queries in Dec' is good enough. To make a precise statement we need the following definitions.

Definition 2.2.10.

- In an execution of Dec' we write *round* i to refer to the step of the computation performed in the simulation of Dec when its i-th query was given to the oracle.
- In an execution of $\mathcal{IBE}'$ with oracle $\mathcal{O}$, we denote by $\mathcal{O}'_i$ the collection of query-answer pairs (excluding those made only at step (2)) throughout the execution of the experiment and up to round i of Dec' (note that $\mathcal{O}'_i$ is different than T_i). That is, the queries made by $\mathsf{Setup}', \mathsf{Keygen}', \mathsf{Enc}'$ and in the first i rounds of Dec'. When no confusion arises we write $\mathcal{O}'$.
- We say that $\mathcal{O}'$ is a *partial group oracle*, or partial oracle, to refer to the fact that $\mathcal{O}'$ can be extended to a group oracle.
- Consider $\mathcal{O}'$ and let $cl(\mathcal{O}')$ to denote its closure, so as to include the answer to every possible query that can be deduced from $\mathcal{O}'$. We say that *an answer to a query* q *is consistent to* $\mathcal{O}'$ if this is the answer to q in $cl(\mathcal{O}')$.

Lemma 2.2.11. *Conditioned on* $\overline{\text{BAD}}$ *every query answered at step (3) is a new query. Also, every query answered at step (1) or (2) is consistent* $\mathcal{O}'$.

Lemma 2.2.11 can be rephrased as follows: Conditioned on $\overline{\text{BAD}}$, $\mathcal{O}'$ is a partial group oracle. This lemma deals only with one part of the theorem. The other has to do with the fact that $\mathcal{O}'$ is properly distributed, and the argument follows.

$\mathcal{IBE}$ and $\mathcal{IBE}'$ behave identically almost everywhere - The proof of Theorem 2.2.5

We bound from below the probability that $\mathcal{IBE}'$ is correct. We first argue that the following is true.

Lemma 2.2.12. $\Pr_{\mathsf{Exp}'}[\mathcal{IBE}' \text{ is correct} \mid \overline{\text{BAD}}] = \Pr_{\mathsf{Exp}}[\mathcal{IBE} \text{ is correct} \mid \overline{\hat{\text{BAD}}}]$, *where* $\hat{\text{BAD}} \subseteq \mathsf{Exp}$ *is an event of the same measure as* $\text{BAD} \subseteq \mathsf{Exp}'$.

Recall that Exp and Exp' are equal as probability spaces - the notation is used to emphasize that the $\mathcal{IBE}$ experiment is over Exp, whereas the $\mathcal{IBE}'$ experiment is over Exp'.

Lemma 2.2.12 follows by a detailed inspection on the probability spaces Exp and Exp'. The main point is that $\mathcal{IBE}'$ is correct for the same fraction of transcripts $\overline{\text{BAD}}$ as $\mathcal{IBE}'$ is correct

for $\overline{\hat{\text{BAD}}}$. We construct a mapping injecting $\overline{\text{BAD}}$ to Exp (the image of $\overline{\text{BAD}}$ is $\overline{\hat{\text{BAD}}}$) with the property that each element of $\overline{\text{BAD}}$ is mapped to an element of $\overline{\hat{\text{BAD}}}$ with the same bit to encrypt and the same bit to decrypt. Let us consider an instance of the security experiment $\eta \in \overline{\text{BAD}} \subseteq \mathsf{Exp}'$. In what follows from η we only list: (i) the possible answers to new queries in Dec' and the additional randomness that Dec' is using when making up new answers at step (3). This suffices for our purposes, since the previously used randomness, and answers to new queries for $\mathsf{Setup}', \mathsf{Keygen}'$ and Enc' will be the same for $\mathcal{IBE}'$ and $\mathcal{IBE}$. Then, the suffix of η is

$$\langle \underbrace{u_1, u_2, \ldots, u_{n^{k'}}}_{\text{answers to new queries}} , \underbrace{u'_1, \ldots, u'_{n^k}}_{\text{randomness for } \mathsf{Dec}'} \rangle$$

Consider an execution of $\mathcal{IBE}'$ on η. By Lemma 2.2.11 we know that the randomness of Dec' is only used to make up answers to new queries. Here is an example.

Example 2.2.13. Say that initially u_1, u_2, u_3 are used for new answers at step (1). For simplicity let us ignore all possible answers to new queries at step (2); in particular, reorder the numbering of u_i's and consider these queries to be at the end. Then, suppose that u'_1, u'_2 are used subsequently for answering new-queries at step (3). And finally suppose that u_4 is used to answer the last new query of Dec'. Now observe, that all the elements of $\overline{\text{BAD}}$ of the following form, result in the same execution for $\mathcal{IBE}'$:

$$\langle \underbrace{u_1, u_2, u_3, u_4, *, *, \ldots, *}_{\text{answers to new queries}}, \underbrace{u'_1, u'_2, *, *, \ldots, *}_{\text{randomness for } \mathsf{Dec}'} \rangle$$

, where for $*$ we chose any string in $\{0,1\}^n$ given that they are distinct. Now, permute the above sequence constructing the following.

$$\langle \underbrace{u_1, u_2, u_3, u'_1, u'_2, u_4, \ldots, *}_{\text{answers to new queries}}, \underbrace{*, *, *, *, \ldots, *}_{\text{randomness for } \mathsf{Dec}'} \rangle$$

By Lemma 2.2.11 when $\mathcal{IBE}$ runs on such permuted elements of $\overline{\text{BAD}}$ will have identical execution, and in particular in the execution of Dec, as $\mathcal{IBE}'$. Observe that we just permuted the fixed strings u_i, u'_j, and the $*$. In other words, the number of transcripts in Exp' and in Exp are the same that result in the particular execution. □

Remark 2.2.14. The actual proof of Theorem 2.2.5 is not much different than the typical case described in this example. The new queries made at step (3) have distinct answers by the choice of r. There is a small technical issue with new queries that are made at step (2). The main remark for these queries is that their value only becomes relevant to test the predicate: "$\mathcal{O}(q^t) = \mathcal{O}(q^{g \text{ or } t}_{old} + \sum_{i=1}^{M} \kappa_i E^g_i)$" at this step. That is, any value (means any oracle) for such new queries works. If such a new query made at step (2) appears later on at step (1) then we

are not going to consider it as a special one, but rather we treat it as one of the u_i's in the above example.

Proof of Theorem 2.2.5. Consider an element $\eta \in \overline{\text{BAD}}$ which provides the parameters and the randomness for the security experiment for $\mathcal{IBE}'$. Record all the new queries made together with those answered at step (3) as in the above example. Apply a permutation to bring it to the form consistent to the execution of $\mathcal{IBE}$ similar to the example. It is immediate by definition that this mapping injects $\overline{\text{BAD}}$ into the probability space of Exp, let $\overline{\hat{\text{BAD}}}$ be the image of this mapping, with the following property: every element of $\overline{\text{BAD}}$ encrypts and decrypts to the same bit as its image in $\overline{\hat{\text{BAD}}}$ (this follows by Lemma 2.2.11). In particular, the number of elements in $\overline{\hat{\text{BAD}}}$ where $\mathcal{IBE}$ is correct is the same as the number of elements in $\overline{\text{BAD}}$.

Therefore, $\Pr_{\mathsf{Exp}'}[\overline{\text{BAD}}] = \Pr_{\mathsf{Exp}}[\overline{\hat{\text{BAD}}}]$ and the number of elements in $\overline{\hat{\text{BAD}}}$ where $\mathcal{IBE}$ is correct, equals the number of elements of $\overline{\text{BAD}}$ where $\mathcal{IBE}'$ is correct.

Now we have the following.

$$\begin{aligned}
\Pr_{\mathsf{Exp}'}[\mathcal{IBE}' \text{ is correct}] &\geq \Pr_{\mathsf{Exp}'}[\mathcal{IBE}' \text{ is correct} \mid \overline{\text{BAD}}] \Pr_{\mathsf{Exp}'}[\overline{\text{BAD}}] \\
&= \Pr_{\mathsf{Exp}}[\mathcal{IBE} \text{ is correct} \mid \overline{\hat{\text{BAD}}}] \Pr_{\mathsf{Exp}}[\overline{\hat{\text{BAD}}}] \\
&= \Pr_{\mathsf{Exp}}[\mathcal{IBE} \text{ is correct} \wedge \overline{\hat{\text{BAD}}}]
\end{aligned}$$

Recall that $\Pr[\overline{\hat{\text{BAD}}}] \geq 1 - f(n)$. Let us denote by Ω the set in the probability space for Exp. That is, $|\overline{\hat{\text{BAD}}}| \geq |\Omega|(1 - f(n))$. Let $\mathcal{E} \subseteq \Omega$ be the event that $\mathcal{IBE}$ is correct. We know that $|\mathcal{E}| \geq \varepsilon|\Omega|$, and let $\varepsilon = \frac{1}{2} + \delta$, $\delta > 0$.

$$\begin{aligned}
|\overline{\hat{\text{BAD}}} \cap \mathcal{E}| = |\overline{\hat{\text{BAD}}}| + |\mathcal{E}| - |\overline{\hat{\text{BAD}}} \cup \mathcal{E}| \geq |\overline{\hat{\text{BAD}}}| + |\mathcal{E}| - |\Omega| \geq |\Omega|(1 - f(n) + \frac{1}{2} + \delta - 1) \\
\implies \Pr_{\mathsf{Exp}}[\mathcal{IBE} \text{ is correct} \wedge \overline{\hat{\text{BAD}}}] \geq \frac{1}{2} + \delta' := \varepsilon'
\end{aligned}$$

for some constant $\delta' > 0$ and sufficiently large n (since $f(n) \to 0$). □

2.2.6 Main Lemmas

In this section we show Lemma 2.2.11. This lemma is a corollary of claims 2.2.17 and 2.2.18. These claims are also used in the upper bound for the probability of BAD in Section 2.2.7 for which we use the following version of Lemma 2.2.11.

We define $\text{BAD}^{(i)}$, $\text{BAD}_1^{(i)}$, $\text{BAD}_3^{(i)}$ as follows.

$\text{BAD}_1^{(i)}$: similar to BAD_1, where the infrequent answer is given in some round $\leq i$.

$\text{BAD}_3^{(i)}$: similar to BAD_3, where the linearly independent vector appears in some round $\leq i$.

Let $\text{BAD}^{(i)} = \text{BAD}_1^{(i)} \vee \text{BAD}_2 \vee \text{BAD}_3^{(i)}$.

Lemma 2.2.15. *Let $\mathfrak{q}$ be the query answered at round i. Conditioned on $\overline{\text{BAD}^{(i)}}$:*

- *If $\mathfrak{q}$ is answered at step (3) then it is a new query.*
- *If $\mathfrak{q}$ is answered at step (1) or (2) then it is consistent to $\mathcal{O}'_i$.*

This lemma follows by claims 2.2.17 and 2.2.18. To show Claim 2.2.17 (below) we observe the following.

Fact 2.2.16. *Let $\mathcal{O}'_i$ be a partial group oracle. At round i if the query $\mathfrak{q}$ is answered at step (1) or (2), then the linear transformations of the formal sums in steps (1) and (2) do not change the discrete logarithm of $\mathfrak{q}$ and $\mathfrak{q}_{old}$.*

Proof. In step (1) we only use the entries of T that contain linear combinations of z_i's and g_i's. These entries are consistent to $\mathcal{O}$. If we rewrite $\mathfrak{q}^z$ as: $\mathfrak{q}^z = \sum_{i=1}^m \alpha_i g_i$, then in particular, $\mathcal{O}(\mathfrak{q}^z) = \mathcal{O}(\sum_{i=1}^m \alpha_i g_i)$, which in particular implies that the discrete logarithm of $\mathfrak{q}^z$ remains the same, and thus the discrete logarithm of $\sum_{i=1}^m \alpha_i g_i + q^t$ is the same as that of $\mathfrak{q}$.

Similarly, in step (2) where we add elements $\mathfrak{q}_{\mathsf{id}}$ whose discrete logarithm is 0. □

Claim 2.2.17. *Let $\mathcal{O}'_i$ be a partial group oracle. Suppose that the query $\mathfrak{q}$ is answered at step (1) or (2).*

- *If $\mathfrak{q}$ is a new query then it is answered (in step (1)) as $\mathcal{O}(\mathfrak{q})$.*
- *Else, if $\mathfrak{q}$ is old, then it is answered as $\mathcal{O}'_i(\mathfrak{q})$, where $\mathcal{O}'_i(q)$ denotes the answer of q from $cl(\mathcal{O}'_i)$.*

This claim follows by Fact 2.2.16.

Claim 2.2.18. *Conditioned on $\overline{\text{BAD}_i}$, if in round $i+1$ a query answered at step (3) then it is a new query.*

To show this claim we use the following two facts.

Let $\mathcal{V}$ be the vector space spanned by the elements of R. Note that the vectors $E_1, \ldots, E_M$ of R are those for which $\mathcal{O}(E_i) = \mathbf{0}$. Let $[q_j] = \mathcal{V} + q_i := \{v + q_i | v \in \mathcal{V}\}$, and $Q := \cup_{I=1}^N [q_i]$, where $q_1, \ldots, q_N$ are the queries in T.

We observe that by the Linear Algebra operations, in step (2) we check whether $q \in Q$.

Fact 2.2.19. *Let $\mathfrak{q}$ be a query such that $\mathcal{O}'_i(q) \in T$. Conditioned on $\overline{\text{BAD}_3^{(i)}}$, step (2) succeeds in answering q at round i.*

Proof. We distinguish between the case that $\mathcal{O}'_i(\mathfrak{q}) = \mathcal{O}(\mathfrak{q})$ and the case where $\mathcal{O}'_i(\mathfrak{q}) \neq \mathcal{O}(\mathfrak{q})$, i.e. the answer $\mathcal{O}'_i(\mathfrak{q})$ was given at step (3).

Case $\mathcal{O}'_i(q) = \mathcal{O}(q)$. We show that if $q \notin Q$, then $T \cup \{q\}$ induces an equality vector linearly independent of R. Since $\mathcal{O}(q) \in T \implies \mathcal{O}(q) = \mathcal{O}(q_j)$ for some $1 \leq j \leq N$. Since $q \notin Q$ i.e. $q \notin [q_j]$ and thus $q \notin \mathcal{V} + q_j \implies (q - q_j) \notin \mathcal{V}$; i.e. $(q - q_j)$ which is linearly independent of the equality vectors in R.

Case $\mathcal{O}'_i(q) \neq \mathcal{O}(q)$. Let $q' = \sum \alpha'_i z_i + \sum \beta'_i t_i$ be the query answered in step (3) and recorded in T with its answer. Let $q = \sum \alpha_i z_i + \sum \beta_i t_i$. Express $\sum(\beta_i - \beta'_i)t_i$ as a linear combination of the g_i's, and consider the vector $q - q'$. We show that is $(\mathfrak{q} - \mathfrak{q}')$ is linearly dependent on R then step (2) succeeds. If $(q - q') \in \mathcal{V}$ then this means that there exists a linear combination $\kappa_1 E_1 + \ldots + \kappa_N E_N = q - q' \implies q' + (\kappa_1 E_1 + \ldots + \kappa_N E_N) = q$. This in particular means that the first IF-statement in step (2) is satisfied. The second IF-statement depends only on the valuation of a vector, which regardless of how we expressed $\sum(\beta_i - \beta'_i)t_i$ if we did it consistently to $\mathcal{O}$ then this IF-statement also succeeds. □

Fact 2.2.20. *Let $\mathfrak{q}'$ be a query made in Enc', and $\mathfrak{q}$ a query made in Dec', such that $\mathcal{O}(q') = \mathcal{O}(q)$. Then, conditioned on $\overline{\text{BAD}_3^{(i)}}$ step (1) succeeds in answering $\mathfrak{q}$ at round i.*

Proof. The queries that Enc' makes are linear combinations of g_i's. The queries that Dec' makes are linear combinations of z_i's and t_i's. Rewrite the linear combination of t_i's as a linear combination of g_i's, as it appears in the execution of Enc'. Since, $\mathcal{O}(q') = \mathcal{O}(q)$ this implies that $\sum \alpha_i z_i = \sum b_i g_i$, which is an equality vector, and since it is not in $\mathcal{V}$ this means it is linearly independent of R. □

Proof of Lemma 2.2.15. Let $\mathfrak{q}$ be a query made by Dec' and suppose that step (3) answers this query. Conditioned on $\overline{\text{BAD}_3^{(i)}}$, $\mathfrak{q}$ cannot have the same answer, and in particular the same discrete logarithm, with a query made in Enc'. This answer to query $\mathfrak{q}$ is not recorded in T, because then conditioned on $\overline{\text{BAD}_3^{(i)}}$, step (2) would have succeeded; i.e. $\mathfrak{q}$ was not made previously in Dec'. Also, by $\overline{\text{BAD}_2^{(i)}}$ we know that the answer to this query cannot be frequent, otherwise we would have recorded it in T. Therefore, $\mathfrak{q}$ has an infrequent answer. By $\overline{\text{BAD}_1}$ this answer is not given anywhere in the other parts of the execution. Thus, $\mathfrak{q}$ is a new query. □

2.2.7 Proof of Lemma 2.2.9

We upper bound each of the probabilities of BAD_2 and BAD.

Bound $\Pr[\text{BAD}_2]$: The probability that in the iterative phase of Keygen' we do not find a frequent answer is $\leq (1 - \frac{1}{n^{7k}})^{n^{7k+2}}) \leq e^{n^2}$. A simple counting argument shows that there are at most polynomially many frequent answers, and by the union bound we have that the

probability that at the end of the execution of Keygen', T contains all frequent answers is $\geq 1 - e^n$. Therefore,

$$\Pr[\text{BAD}_2] \leq e^{-n}$$

Bound $\Pr[\text{BAD}]$: For this we use an inductive argument that relies on Lemma 2.2.15 and an argument similar to the proof of Theorem 2.2.5.

Let us start with some elementary observations regarding the filtration of $\overline{\text{BAD}^{(i)}}$. Unless mentioned otherwise all probabilities are over Exp'. Observe that $\text{BAD}^{(1)} \subseteq \text{BAD}^{(2)} \subseteq \ldots$, and similarly for $\text{BAD}_1^{(i)}$ and $\text{BAD}_3^{(i)}$. From this and by the definition of conditional probability we have $\mathsf{Pr}[\overline{\text{BAD}^{(i+1)}}] = \mathsf{Pr}[\overline{\text{BAD}^{(i+1)}}|\overline{\text{BAD}^{(i)}}]\mathsf{Pr}[\overline{\text{BAD}^{(i)}}]$. Hence, by inducting on m we obtain:

$$\mathsf{Pr}[\overline{\text{BAD}^{(m)}}] = \mathsf{Pr}[\overline{\text{BAD}^{(m)}}|\overline{\text{BAD}^{(m-1)}}] \cdot \mathsf{Pr}[\overline{\text{BAD}^{(m-1)}}|\overline{\text{BAD}^{(m-2)}}] \ldots \mathsf{Pr}[\overline{\text{BAD}^{(2)}}|\overline{\text{BAD}^{(1)}}] \cdot \mathsf{Pr}[\overline{\text{BAD}^{(1)}}]$$

where $m = n^k$. By definition $\mathsf{Pr}[\overline{\text{BAD}^{(m)}}] = \mathsf{Pr}[\overline{\text{BAD}}]$, and thus it suffices to lower bound the above. Therefore, it is sufficient to lower bound $\mathsf{Pr}[\overline{\text{BAD}^{(1)}}]$ and for an arbitrary i to lower bound $\mathsf{Pr}[\overline{\text{BAD}^{(i+1)}}|\overline{\text{BAD}^{(i)}}]$. By Lemma 2.2.15 we have that up to round i, $\mathcal{O}'_i$ is a partial group oracle. Therefore, using a very similar (almost identical) argument as in the proof of Theorem 2.2.5 we can show that

$$\mathsf{Pr}_{\mathsf{Exp}'}[\overline{\text{BAD}^{((i+1))}} \mid \overline{\text{BAD}^{(i)}}] = \mathsf{Pr}_{\mathsf{Exp}}[\overline{\hat{\text{BAD}}^{(i+1)}} \mid \overline{\hat{\text{BAD}}^{(i)}}]$$

By definition of these events, and the definition of conditional probability we obtain:

$$\mathsf{Pr}_{\mathsf{Exp}}[\hat{\text{BAD}}^{(i+1)} \mid \overline{\hat{\text{BAD}}^{(i)}}] \leq \mathsf{Pr}_{\mathsf{Exp}}[\hat{\text{BAD}}_1] + \mathsf{Pr}_{\mathsf{Exp}}[\hat{\text{BAD}}_2] + \mathsf{Pr}_{\mathsf{Exp}}[\hat{\text{BAD}}_3]$$

where $\hat{\text{BAD}}_1$, $\hat{\text{BAD}}_2$, $\hat{\text{BAD}}_3$ are defined similarly to the hatted events in the proof of Theorem 2.2.5. Therefore, it is sufficient: to bound the probability of (i) the event $\hat{\text{BAD}}_1$, that there is an infrequent answer in the execution of Dec, and of (ii) the event $\hat{\text{BAD}}_3$, that there is any new linearly independent equality vector in the last $\mathsf{Enc\text{-}Dec}$ execution, *in* Exp. By (ii) we mean that: there is a polynomial number of $\mathsf{Enc\text{-}Dec}$ executions in the iterative phase of Keygen, and the last one is the execution of the actual security experiment.

We first bound $\mathsf{Pr}_{\mathsf{Exp}}[\hat{\text{BAD}}_1]$. There are at most $3n^k$ distinct answers in $\mathsf{Setup}', \mathsf{Enc}'$ and the simulation of Keygen by Keygen'. Also, there are at most n^k distinct answers in Dec' (excluding step (2)). By taking the union bound we have that: $\mathsf{Pr}_{\mathsf{Exp}}[\hat{\text{BAD}}_1] < \frac{3}{n^{3k}}$.

Now, we bound $\mathsf{Pr}_{\mathsf{Exp}}[\hat{\text{BAD}}_3]$. The iterative phase of Keygen takes a random number of iterations between n^{7k+2} and $n^{7k+2} + 2n^{4k}$. Each iteration is an independent execution of the pair of algorithms $\mathsf{Enc\text{-}Dec}$. Since each equality vector is a formal sum of $2n^k$ terms this means that the dimension of the space spanned by the equality vectors of R is $\leq 2n^k$. Consider all the independent executions together with the last execution which is the actual execution of

the security experiment. There are at most $2n^k$ new equality vectors that appear between n^{7k+2} and $n^{7k+2} + 2n^{4k}$ executions. Since the number of executions is chosen at random the probability that we actually hit an execution where an equality vector appears is $\leq \frac{2n^k}{2n^{4k}} = \frac{1}{n^{3k}}$. Therefore,

$$\mathsf{Pr}_{\mathsf{Exp}}[\hat{\text{BAD}}_3] \leq \frac{1}{n^{3k}}$$

That is, $\mathsf{Pr}[\overline{\text{BAD}^{(i+1)}}|\overline{\text{BAD}^{(i)}}] \geq 1 - (\frac{1}{n^{3k}} + e^{-n} + \frac{3}{n^{3k}}) \geq 1 - \frac{1}{n^{2k}}$. Furthermore, $\mathsf{Pr}[\overline{\text{BAD}^{(1)}}] \geq 1 - (e^{-n} + \frac{1}{n^{3k}} + \frac{3}{n^{3k}}) \geq 1 - \frac{1}{n^{2k}}$.

Putting everything together, we have that (recall that $m = n^k$)

$$\mathsf{Pr}[\overline{\text{BAD}}] = \mathsf{Pr}[\overline{\text{BAD}^{(n^k)}}] \geq (1 - \frac{1}{n^{2k}})^{n^k - 1}(1 - \frac{1}{n^{2k}}) \implies \mathsf{Pr}[\text{BAD}] \leq f(n)$$

where $f : \mathbb{Z}^+ \to [0, 1]$ monotonically decreases to 0. □

2.3 On the impossibility of constructing a PRG that makes non-adaptive use of a OWP

The content of this section has been significantly generalized and extended in [JP10].

We show that it is not possible to base the construction of a PRG of super-linear stretch on a non-adaptive Goldreich-Levin-like construction; i.e. by relying in a black-box and non-adaptive manner on an OWP π.

Consider the following scenario for motivation. We wish to compute a PRG G of super-linear stretch in a *streaming manner*, i.e. (i) using small space and (ii) reading the seed in the input a small number of times. Say that "small" means $\log^{O(1)} n$, for a seed of length n. Let us also make the simplifying assumption (which only makes lower bounds stronger) that it is possible to compute π using small space and number of passes over the input. We intuitively argue that, in at least two ways, adaptive use of π is not possible in a streaming setting.

1. π is reasonably hard[12], e.g. $n^{\log n}$-hard. Then, a natural way of computing a secure PRG using π, requires applying π on strings of length n^ϵ. Recall that G is computed under very limited space-passes, and the seed is random and in particular incompressible. A natural way of making adaptive use of π requires storing intermediate parts of the computation or at least recomputing them. Recall that we lack of space (so as to store) and access to the input (so as to recompute).

2. The algorithm works in space $O(\log n)$. Then, regardless how hard π is, "in place" adaptive use of π is out of the question.

[12] On the other hand, if π were very hard *and* the algorithm had say $\log^2 n$ working space, then it would have been possible to perform the adaptive computation "in place" in the working memory (see e.g. Example 5.1.1).

Notational conventions. As usual, non-integer values are rounded up. Let $m \in \mathbb{Z}^+$, then $[m] = \{1, \ldots, m\}$. We denote by $\mathsf{neg}(n) : \mathbb{Z}^+ \to \mathbb{R}^+$ a *negligible function*, i.e. $\mathsf{neg}(n) < n^{-k}$, for every $k \in \mathbb{Z}^+$ and sufficiently large $n \in \mathbb{Z}^+$. Let $m \in \mathbb{Z}^+$, $a \in [m]$, and $n \in \mathbb{Z}^+$. Then, we denote by $\langle a \rangle_n$ the n-bit binary string representation of a, padded with trailing zeros when necessary. If it is clear from the context, e.g. in $a \in [m]$, then we write $\langle a \rangle$ instead of $\langle a \rangle_{\lceil \log m \rceil}$. If $a > 2^n$, then $\langle a \rangle_n$ denotes the n least significant bits of the binary representation of a.

Definition 2.3.1. Let $\alpha > 1$, $G^{(\cdot)} : \{0,1\}^{2n} \to \{0,1\}^{n+n^\alpha}$ be a number generator with oracle access. $G^{(\cdot)}$ is *bitwise non-adaptive* if there exist computable functions $f = \left\{ f_n : \{0,1\}^n \times [n^\alpha] \to \{0,1\} \right\}$, and $B = \left\{ B_n : \{0,1\}^{3n} \times [n^\alpha] \to \{0,1\} \right\}$ such that for every $n \in \mathbb{Z}^+, r, x \in \{0,1\}^n$ we have

$$G^{(\cdot)}(r, x) = r, b_1, \ldots, b_{n^\alpha}$$

where $b_i := B_n(r, x, \pi(f_n(x, i)), i)$, $i = 1, \ldots, n^\alpha$.

Note that in this definition f, B are computationally unbounded. We only consider permutations $\{0,1\}^* \to \{0,1\}^*$, such that a permutation $\pi : \{0,1\}^* \to \{0,1\}^*$ can be identified by $\pi = \{\pi_n : \{0,1\}^n \to \{0,1\}^n\}_n$, where each π_n is a permutation.

Here is the black-box separation theorem.

Theorem 2.3.2. *Let $\alpha > 1$ and let f, B as in Definition 2.3.1; i.e. $G^{(\cdot)} : \{0,1\}^{2n} \to \{0,1\}^{n+n^\alpha}$ is a bitwise non-adaptive number generator. Then, there exists a distribution Π over permutations such that*

(i) there exists a uniform adversary Adv so that with probability 1 over a randomly chosen $\pi \in \Pi$, Adv breaks the pseudo-randomness of G^π with oracle access and $n^{O(1)}$ many queries to π

(ii) with probability 1 over a randomly chosen $\pi \in \Pi$, π is one-way against computationally unlimited uniform adversaries that make polynomially many queries, with oracle access to π.

In the language of [RTV04], Theorem 2.3.2 shows that there is no fully black-box reduction of the pseudorandomness of G^π to the one-wayness of π. Although in a different context, our proofs are inspired by the techniques of [GGKT05].

In the next Section 2.3.1 we give some intuition regarding the high-level of the proof. In Section 2.3.2 we give the proof of Theorem 2.3.2.

2.3.1 Discussion for Theorem 2.3.2

Here is an example that puts in perspective Theorem 2.3.2. In this example the function f is efficiently computable and easy to invert. Note that if we were aiming at a stronger form of black-box separation, then one of the requirements would be that f alone (without using π) does not imply the existence of a PRG.

Example 2.3.3. Let B_n be the the usual inner product over $\mathsf{GF}(2)^n$, $\langle x, y\rangle$, $x, y \in \mathsf{GF}(2)^n$, as in the Goldreich-Levin construction. Here is a simple $f : \{0,1\}^n \times \{0,1\}^{n^\alpha} \to \{0,1\}^n$. Define $f(x,i) = \langle \bar{x} + i\rangle_n$, where $\bar{x}$ denotes the integer with binary representation x. Let us relax for a moment the requirement that π_n will be a permutation. Partition $\{0,1\}^n$ into the disjoint sets $S_1, S_2, \ldots, S_m$, where $m = 2^n/n^{\alpha^2}$. S_1 is the set of n-bit binary representations of the integers $\{0, \ldots, n^{\alpha^2} - 1\}$, S_2 is identified by $\{n^{\alpha^2}, \ldots, 2n^{\alpha^2} - 1\}$ and so on. Let $S = \{S_1, \ldots, S_m\}$. We define a distribution over functions $\{0,1\}^n \to \{0,1\}^n$ as follows. Define the distribution $\Pi_n = \{\pi_n\}$ by choosing a random permutation $\tau : S \to S$, and for each S_i map each one of its elements to the smallest integer in $\tau(S_i)$. That is, for a randomly chosen $x \in \{0,1\}^n$ and π_n, with probability $\geq 1 - \frac{1}{n^{O(1)}}$ we have that $\pi_n(f(x,1)) = \pi_n(f(x,2)) = \cdots = \pi_n(f(x,n^\alpha))$. The test for the adversary is whether $b_1 = b_2 = \ldots = b_{n^\alpha}$. On the other hand the set S is very large so that by standard arguments (e.g. [IR88]) π_n is one-way.

It takes little to generalize this example to work for one-way permutations instead of functions. Instead of mapping every element of S_i to the same element in $\tau(S_i)$ we map the j-th smallest element of S_i to the j-th smallest element in $\tau(S_i)$. Observe that with constant probability the say last 3 bits of r are all zero, and thus with a sufficiently large constant probability $\langle r, \pi_n(f(x,1))\rangle = \langle r, \pi_n(f(x,2))\rangle = \cdots = \langle r, \pi_n(f(x,7))\rangle$ which allows us to distinguish between random and pseudorandom strings. □

In this example f has very special structure. The adversarial construction of the distribution depends on the fact that f is the addition of $\bar{x}$ and i. Theorem 2.3.2 refers to arbitrary computable f's (and B's) but also to computationally unbounded adversaries. It becomes more technical to generalize the above example so as to make a computationally unbounded adversary to deal with f's whose *range-preimage size* is small; where by this we mean that for every $\sigma = (x,i) \in \{0,1\}^n \times [n^\alpha]$ we have $|f^{-1}(f(\sigma))| \leq n^{O(1)}$.

On the other extreme we have to deal with f's where the range-preimage is superpolynomial, i.e. $|f^{-1}(f(\sigma))| > n^k$, for every k and sufficiently large n. But, in this case by simple counting we have that the image of f is of negligible size; i.e. $\frac{f|(\{0,1\}^n \times [n^\alpha])|}{2^n} = \mathsf{neg}(n)$. Therefore, we can define the distribution on the permutations π_n to be a random permutation on the complement of the image of f, and define π_n on $f(x,1), f(x,2), \ldots$ arbitrarily.

An issue occurs when the range-preimage size on inputs $\sigma = (x, i)$ is neither polynomial nor super-polynomial. The argument in the proof of Theorem 2.3.2 takes into account a more appropriate statistic on the range-preimage size, and deals with the case where f_n, B_n are arbitrary.

2.3.2 Proof of Theorem 2.3.2

Notation. Let $\alpha > 1$. Let $D_{n,\alpha} = \{0,1\}^n \times [n^\alpha]$, $N_{n,\alpha} := |D_{n,\alpha}| = n^\alpha 2^n = 2^{n+\alpha \log n}$. When no confusion arises, we instead write D, N. Occasionally, we identify $D = \{0,1\}^n \times [n^\alpha]$ by $\{0,1\}^{n+\alpha \log n}$ in the usual way. By x, r we denote the two n-bit arguments to G^π; i.e. $x, r \in \{0,1\}^n$. Recall that the notation $\langle i \rangle_m w \in \{0,1\}^n$ denotes an n-bit string, which is the concatenation of $\langle i \rangle_m \in \{0,1\}^m$, and $w \in \{0,1\}^{n-m}$; and furthermore i is an integer $i \in [2^m]$. Let $f_n : D \to \{0,1\}^n$ a function. For $y \in \{0,1\}^n$, the *preimage size of* y (with respect to f) is denoted by $s^f(y) = |f^{-1}(y)|$. For $\sigma = (x, i) \in D$, the *range-preimage size of* σ (with respect to f) is denoted by $t^f(\sigma) = |f^{-1}(f(\sigma))|$. Extend t^f to sets $A \subseteq D$, $t^f(A) = \{t \mid t = t^f(\sigma), \sigma \in A\}$. The *median range-preimage size of* f is

$$\mathsf{Med}_n(f) = \min_{A \subseteq D, |A| \geq N/2} \max t^f(A)$$

That is, consider the multiset consisting of $t^f(\sigma)$ for every $\sigma \in D$ and denote by $\mathsf{Med}_n(f)$ its median (if N is even then $\mathsf{Med}_n(f)$ is the next to median).

We show Theorem 2.3.2 by considering two cases. The first, and simpler one, is when $\mathsf{Med}_n(f)$, which is a function of n, is superpolynomial for sufficiently large n. The second is when $\mathsf{Med}_n(f)$ is bounded by a polynomial for infinitely many n. In each of the two cases we show that there is a distribution over the permutations π such that with probability 1, a chosen π is one-way, whereas there exists an adversary such that with probability 1 breaks the pseudorandomness of G^π.

Case: $\mathsf{Med}_n(f) > n^k$, for every $k \in \mathbb{Z}^+$ and sufficiently large n

This is the simpler case. Fix $\alpha > 1$, $f = \{f_n : D_{n,\alpha} \to \{0,1\}^n\}_n$. For every $n \in \mathbb{Z}^+$, let $\mathsf{Big}_{f_n} = \{\sigma \mid t^{f_n}(\sigma) \geq \mathsf{Med}_n(f)\} \subseteq D$. We simply write Big when f_n is understood from the context. There are three facts regarding sizes of sets on which our argument is based.

1. $|\mathsf{Big}_{f_n}|$ is large with respect to N, and in particular $\frac{1}{2}N$.
2. There exists a large set of x's, $x \in \{0,1\}^n$, such that for every such x there is a constant fraction of i's, $i \in [n^\alpha]$, for which $(x, i) \in \mathsf{Big}_{f_n}$.
3. $|f_n(\mathsf{Big}_{f_n})|$ is of negligible size with respect to N.

We will define the distribution over permutations baring in mind all three bullets. The facts in bullets (1) and (2) will be used in arguing that our adversary works, whereas the fact in bullet (3) will be used in the proof that a random member of the defined distribution is one-way.

Let us argue for each of these three facts.

By definition, we have $|\mathsf{Big}| \geq \frac{1}{2}N$.

For every $n \in \mathbb{Z}^+$, let $\mathsf{Good} \subseteq \{0,1\}^n$ be the set of $x \in \{0,1\}^n$ for which there are at least $\frac{1}{4}n^\alpha$ many i's such that $(x,i) \in \mathsf{Big}$. Therefore, by counting $|\mathsf{Good}| \geq \frac{1}{4}2^n$. The adversary Adv will be defined such that it cares to distinguish between random and pseudorandom strings, when the pseudorandom strings come from $x \in \mathsf{Good}$ (recall that Good is sufficiently large).

Claim 2.3.4. *$|f_n(\mathsf{Big})| < \frac{2^n}{n^d}$, for every $d \in \mathbb{Z}^+$ and sufficiently large n.*

Proof. Suppose that there exists d such that $|f_n(\mathsf{Big})| \geq \frac{2^n}{n^d}$ for infinitely many n. By the general assumption we have that for $k = d + \alpha + 1$ and for sufficiently large n, $\mathsf{Med}_n(f) > n^{d+\alpha+1}$. By definition, for every $y \in f_n(\mathsf{Big})$ we have that $s^{f_n}(y) \geq \mathsf{Med}_n(f) > n^{d+\alpha+1}$. Therefore, for infinitely many n we have that

$$|\mathsf{Big}| \geq n^{d+\alpha+1}|f_n(\mathsf{Big})| \geq n^{d+\alpha+1}\frac{2^n}{n^d} = 2^{n+(\alpha+1)\log n} > N$$

which is a contradiction. □

We now define the distribution $\Pi = \{\Pi_n\}$ on the permutations, where we can break with probability 1 the pseudorandomness of G^π, and at the same time a randomly chosen π is one-way with probability 1.

Distribution $\Pi = \{\Pi_n\}_{n\in\mathbb{Z}^+}$: Let $n \in \mathbb{Z}^+$. We define $\pi_n \in \Pi_n$ as follows. π_n is the identity on $f_n(\mathsf{Big})$, and it is a random permutation outside $f_n(\mathsf{Big})$; i.e. π_n is determined by choosing a uniformly random permutation $\big(\{0,1\}^n - f_n(\mathsf{Big})\big) \to \big(\{0,1\}^n - f_n(\mathsf{Big})\big)$.

We define the adversary Adv for inputs of length $n + n^\alpha$, $n \in \mathbb{Z}^+$, as follows.

Adversary Adv. Input: $y = r, b_1, \ldots, b_{n^\alpha}$, $r \in \{0,1\}^n, b_i \in \{0,1\}$: The adversary accepts iff there exists $x \in \mathsf{Good}$ such that for the $\frac{1}{4}n^\alpha$ many smallest i's, $i \in [n^\alpha]$, where $(x,i) \in \mathsf{Big}$ we have that $b_i = B_n(r, x, f_n(x,i), i)$.

Claim 2.3.5. *For every π_n in the support of Π_n the adversary Adv breaks the pseudorandomness of G^{π_n}, for sufficiently large $n \in \mathbb{Z}^+$. In particular, with probability 1 over the choice of π, Adv breaks the pseudorandomness of G^π.*

Proof. Let $n \in \mathbb{Z}^+$ and $\pi_n \in \Pi_n$.

If $x \in \mathsf{Good}$ then $\mathsf{Adv}(G^{\pi_n}(r,x))$ accepts. Therefore, for randomly chosen $r, x \in \{0,1\}^n$, the adversary Adv accepts with probability $\geq \frac{1}{4}$.

We show that given a random input y the probability that Adv accepts is inverse exponentially small, and this is sufficient to conclude the lemma. Let $y = r, b_1, \ldots, b_{n^\alpha} \in_R \{0,1\}^{n+n^\alpha}$, where $r \in \{0,1\}^n$ and $b_i \in \{0,1\}$. Recall that a necessary condition for Adv to accept is that there exists an $x \in \mathsf{Good}$ that results in the b_i. Let us count the number of distinct $\langle b'_1, \ldots, b'_{n^\alpha}\rangle$ that pass the test of Adv. For every $r' \in \{0,1\}^n$ and every $x \in \mathsf{Good}$ there are $2^{\frac{3}{4}n^\alpha}$ strings $\langle b'_1, \ldots, b'_{n^\alpha}\rangle$ where for the $\frac{1}{4}n^\alpha$ many smallest i's where $(x,i) \in \mathsf{Big}$ we have that $b'_i = B_n(r', x, f_n(x,i), i)$. Therefore, for every $r' \in \{0,1\}^n$ there are at most $|\mathsf{Good}|2^{\frac{3}{4}n^\alpha} \leq 2^n 2^{\frac{3}{4}n^\alpha} = 2^{\frac{3}{4}n^\alpha + n}$ strings $\langle b'_1, \ldots, b'_{n^\alpha}\rangle$ where Adv accepts. Since $\alpha > 1$, for a randomly chosen y, Adv accepts with probability $\leq 2^{\frac{3}{4}n^\alpha + 2n}/2^{n^\alpha + n} = 1/2^{\frac{1}{4}n^\alpha - n} \leq 2^{-n}$, for sufficiently large n. □

We conclude the proof for this case by showing that π is one-way against uniform, probabilistic adversaries, with oracle access to π and Adv.

Remark 2.3.6. Recall that the oracles in Sections 2.1 and 2.2 are related in a very natural way with the TDP and DDH primitives respectively. In these cases the argument that the primitives exist is self-evident. On the other hand, in this section we specifically tailor the distribution over the permutations, and to that end a more careful treatment is needed.

As usual we are going to show that π is one-way even for computationally unbounded adversaries. To that end, we follow the standard approach, introduced in [IR88].

Lemma 2.3.7. *With probability* 1 *over the choice of* $\pi \in \Pi$, π *is one-way against computationally unbounded Turing Machines that make* $n^{O(1)}$ *queries to* π.

Proof. Let $M^{(\cdot)}$ be a computationally unbounded oracle Turing Machine that on inputs of length n makes $\leq n^c$ queries to the oracle, for some $c > 0$. First let us assume that M makes queries only of length n. We show that for every input length n, $M^{(\cdot)}$, on input $\pi_n(x)$, for randomly chosen $\pi_n \in \Pi_n$ and $x \in \{0,1\}^n$, cannot output x with probability better than $\mathsf{neg}(n)$, when given oracle access to π_n. Consider $M^{\pi_n}(\pi_n(x))$. Without loss of generality, the output of M^{π_n} is one of its oracle queries. We upper bound the probability that M^{π_n} queries x. Conditioned on $x \notin f_n(\mathsf{Big})$ we have that the probability that M^{π_n} queries x is $\leq \frac{n^c}{2^n - |f_n(\mathsf{Big})|}$. Therefore, the probability that M queries x is

$$\mathsf{Pr}[M^{\pi_n}(\pi_n(x)) \text{ queries } x] \leq \frac{2^n - |f_n(\mathsf{Big})|}{2^n} \frac{n^c}{2^n - |f_n(\mathsf{Big})|} + 1\frac{|f_n(\mathsf{Big})|}{2^n}$$

By Claim 2.3.4, we have that this probability is $\leq \frac{1}{n^{d'}}$ for every $d' > 0$ and sufficiently large n.

We now consider M with oracle access to π, i.e. M^π is allowed to make queries to different lengths. Then, the probability in the above calculation can only become smaller when the machine is wasting its polynomially many queries for other input lengths.

Therefore, for every $d'' > 0$ and sufficiently large n, the measure of $\pi \in \Pi$ such that

$$\Pr_{x\in\{0,1\}^n}[M^\pi(\pi(x)) = x] \geq \frac{1}{n^{d''}}$$

is less than $1/n^2$. Therefore, for every $d'' > 0$, by the Borel-Cantelli lemma we have that the measure of $\pi \in \Pi$ such that $\Pr_{x\in\{0,1\}^n}[M^\pi(\pi(x)) = x] \geq \frac{1}{n^{d''}}$ for infinitely many n is 0.

Since there are only countably many Turing Machines, the measure of π such that there exists a Turing Machine that makes $n^{O(1)}$ oracle queries and breaks the one-wayness of π is 0. □

Case: $\mathsf{Med}_n(f) \leq n^k$, for some $k \in \mathbb{Z}^+$ and infinitely many n

Let $k \in \mathbb{Z}^+$, $\mathcal{N} \subseteq \mathbb{Z}^+$ be the set of all n such that $\mathsf{Med}_n(f) \leq n^k$. Let $\alpha > 1$ and $f = \{f_n : D_{i,\alpha} \to \{0,1\}^n\}$. Fix $n \in \mathcal{N}$. We now define the distribution on permutations π for which we will break G^π, and at the same time π is one-way.

Let $\mathsf{Small} \subseteq D_{n,\alpha}$ defined as $\mathsf{Small} = \{\sigma | t^{f_n}(\sigma) \leq \mathsf{Med}_n(f), \sigma \in D_{n,\alpha}\}$. Therefore, $|\mathsf{Small}| \geq \frac{N}{2}$. For every $n \in \mathcal{N}$, let $\mathsf{Good} \subseteq \{0,1\}^n$ be the set of $x \in \{0,1\}^n$ such that there are at least $\frac{1}{4}n^\alpha$ many i's such that $(x,i) \in \mathsf{Small}$. Therefore, by counting $|\mathsf{Good}| \geq \frac{1}{4}2^n$. For every $n \in \mathcal{N}$ we define a set $\mathsf{Good}' \subseteq \{0,1\}^n$, a permutation $h : \{0,1\}^n \to \{0,1\}^n$ and a function $\delta : D \to [n^\alpha]$.

Remark 2.3.8. The function h is used to rearrange the image of f_n and related appropriately with δ. In what follows, δ, h and Good are defined together. Immediately after the description of the algorithm we list its properties, relevant to the proof.

Initial conditions: Good as before, $\mathsf{Good}' \leftarrow \emptyset$, h and δ undefined.

1. While $\mathsf{Good} \neq \emptyset$ do:
1.1 Let $x \in \mathsf{Good}$ be lexicographically smallest, and let $x = \langle v \rangle w$, $v \in [n^\alpha]$, $w \in \{0,1\}^{n-\alpha\log n}$
1.2 Let $I \leftarrow (i_1, \ldots, i_{n^\alpha})$ be the sequence of the $\frac{1}{4}n^\alpha$ smallest i's, $i_j \in [n^\alpha]$, in increasing order such that $(x,i) \in \mathsf{Small}$
1.3 For every $j \leftarrow 1, \ldots, \frac{1}{4}n^\alpha$ do:
1.3.1 Let $\delta(x, i_j) \leftarrow i$, where i is smallest such that $f_n(x,i) = f_n(x,i_j)$.
1.3.2 If $\delta(x,i_j) = i_j$ then
Let $h(f_n(x,i_j)) \leftarrow \langle i_j \rangle w$

For every $x' \in \mathsf{Good}$ where there exists $i' \in [n^\alpha]$ such that
$(x', i') \in \mathsf{Good}$ and $f_n(x', i') = f_n(x, i_j)$, then remove x' from Good

1.3.3 For every $x' \in \mathsf{Good}$ where there exists $i' \in [n^\alpha]$ where $x' = \langle i' \rangle w$, then remove x' from Good

1.3.4 Add x to Good'

2. For every $x \in \{0,1\}^n$ where h is undefined, assign an arbitrary unused value from $\{0,1\}^n$ to $h(x,i)$. For every $x \in \{0,1\}^n$, $i \in [n^\alpha]$ where $\delta(x,i)$ is undefined, let $\delta(x,i) = \perp$.

We claim that the above procedure for constructing $\mathsf{Good}', h, \delta$ is well-defined. The existence of I in (1.2) is guaranteed by definition of Good. Before (2), h is well-defined since in each iteration where $h(y)$ is defined, the elements of Good that can cause problem later on (i.e. by redefining $h(y)$) are removed from Good. It is also easy to see that before (2), h is injective. Therefore, at step (2) h is well-defined as well, and at the end h is a permutation $\{0,1\}^n \to \{0,1\}^n$. The places where δ is $\perp$ will not be of relevance to the argument. Finally, we observe that the constructed Good' is of noticeable size, as stated below.

The above procedure constructs $\mathsf{Good}', h, \delta$ with the following properties which follow by an easy induction.

(i) Good' is large enough (Claim 2.3.9).

(ii) $h : \{0,1\}^n \to \{0,1\}^n$ is a bijection; i.e. h permutes $\{0,1\}^n$.

(iii) Let $x \in \mathsf{Good}'$ and I_x be the set of the $\frac{1}{4}n^\alpha$ many i's used in the above procedure; i.e. in particular, for every such i, $(x,i) \in \mathsf{Small}$.

(iv) For every $x \in \mathsf{Good}'$ and $i \in I_x$, we have that $h(f_n(x,i)) = \delta(x,i)\mathsf{suffix}(x)$, where $\mathsf{suffix}(x)$ is the $n - \alpha \log n$ bit long suffix of x.

Claim 2.3.9. *For every $n \in \mathcal{N}$, when the procedure terminates $|\mathsf{Good}'| \geq 2^n/(n^{\alpha+k} + 4n^\alpha) = 2^n/n^{O(1)}$.*

Proof. In each of the $\geq \frac{1}{4}2^n$ iterations of loop (1) one element is added to Good' and at most $\frac{1}{4}n^\alpha + n^\alpha$ are removed. This is because, in each of the $\frac{1}{4}n^\alpha$ iterations of the loop (1.3) at most n^k elements are removed, and by (1.3.3) at most n^α elements are removed. □

We define the distribution $\Pi = \{\Pi_n\}$, where Π_n is a distribution over permutations $\pi_n : \{0,1\}^n \to \{0,1\}^n$, $n \in \mathcal{N}$.

Distribution $\Pi = \{\Pi_n\}$:

- $\Pi' = \{\Pi'_n\}$: the uniform distribution.

- $\Pi'' = \{\Pi''_n\}$: determine a π''_n as follows. First choose $\pi'_{\log^2 n} \in \Pi'_{\log^2 n}$, this will be associated with the last $\log^2 n$ bits. Define $\pi''_n(x_1x_2) = x_1\pi'_n(x_2)$, where $x_1 \in \{0,1\}^{n-\log^2 n}, x_2 \in \{0,1\}^{\log^2 n}$.

- $\Pi = \{\Pi_n\}$: let $\pi_n = \pi''_n \circ h$, where π''_n is chosen from Π''_n (as usual, $\pi''_n \circ h(x) := \pi''_n(h(x))$).

We define the adversary Adv for inputs of length $n + n^\alpha$, $n \in \mathcal{N}$, as follows.

Adversary Adv. Input: $y = r, b_1, \ldots, b_{n^\alpha}, \quad r \in \{0,1\}^n, b_i \in \{0,1\}$**:** The adversary accepts iff there exists $x \in \mathsf{Good}'$ and $z \in \{0,1\}^{\log^2 n}$ such that by defining $j \in [n^\alpha]$, $v \in \{0,1\}^{n-\alpha\log n-\log^2 n}, w \in \{0,1\}^{\log^2 n}$ so that $x = \langle j\rangle vw$, $i \in I_x$ are such that $b_i = B_n(r, x, \delta(x,i)vz, i)$.

Claim 2.3.10. *For every $n \in \mathcal{N}$ and every π_n in the support of Π_n the adversary Adv breaks the pseudorandomness of G^{π_n}. In particular, with probability 1 over the choice of π, Adv breaks the pseudorandomness of G^π.*

Proof. Fix $n \in \mathcal{N}$ and π_n from the support of Π_n.

If $x \in \mathsf{Good}'$ and for every $r \in \{0,1\}^n$, then by definition $\mathsf{Adv}(G^{\pi_n}(r,x))$ accepts. That is, Adv accepts with probability $\geq \frac{1}{n^{\alpha+k}+4n^\alpha}$.

In what follows we show that when the input y is random, then $\mathsf{Adv}(y)$ accepts with probability exponentially small in n, which is sufficient to conclude the claim. For every $r' \in \{0,1\}^n, x \in \mathsf{Good}'$ and $z \in \{0,1\}^{\log^2 n}$, there are $\leq 2^{\frac{3}{4}n^\alpha}$ many $\langle b'_1, \ldots, b'_{n^\alpha}\rangle$ such that defining $i \in [n^\alpha], v \in \{0,1\}^{n-\alpha\log n-\log^2 n}, w \in \{0,1\}^{\log^2 n}$ so that $x = \langle j\rangle vw$ we have that the $\frac{1}{4}n^\alpha$ many smallest i's where $(x,i) \in \mathsf{Small}$ are such that $b'_i = B_n(r', x, \delta(x,i)vz, i)$. Therefore, for every $r' \in \{0,1\}^n$ there are at most $|\mathsf{Good}'|2^{\log^2 n+\frac{3}{4}n^\alpha} \leq 2^n 2^{\log^2 n+\frac{3}{4}n^\alpha} = 2^{\frac{3}{4}n^\alpha+n+\log^2 n}$ strings $b'_1, \ldots, b'_{n^\alpha}$ where Adv accepts $r', b'_1, \ldots, b'_{n^\alpha}$. Since, $\alpha > 1$, the probability that Adv accepts on a random input is $\leq \frac{2^{\frac{3}{4}n^\alpha+2n+\log^2 n}}{2^{n^\alpha+n}} = \frac{1}{2^{\frac{1}{4}n^\alpha-n-\log^2 n}} \leq 2^{-n}$, for sufficiently large n. □

Similarly to the proof of Lemma 2.3.7, we conclude in this case too, by showing that π is one-way against uniform, probabilistic adversaries, with oracle access to π and Adv. We show that π is one-way even for computationally unbounded adversaries.

Claim 2.3.11. *With probability 1 over the choice of $\pi \in \Pi$, π is one-way against computationally unbounded Turing Machines that make $n^{O(1)}$ queries to π.*

Proof. For every $\pi \in \Pi$, let $P_\pi : \{0,1\}^* \to \{0,1\}^*$ be as follows. For every $n \in \mathcal{N}$ and $x \in \{0,1\}^n$, $P_\pi(x) := \pi(h^{-1}(x))$. For every $n \notin \mathcal{N}$ and $x \in \{0,1\}^n$, $P_\pi(x) := \pi(x)$.

Now, the following is easy to show.

Claim 2.3.12. *If P_π is one-way with respect to probabilistic, computationally unlimited Turing Machines that make $n^{O(1)}$ queries to the oracle, then so is π.*

Therefore, it suffices to show that for randomly chosen $\pi \in \Pi$, P_π is one-way with probability 1. By definition of Π and Π'', if $\pi'' \in \Pi''$ and $\pi \in \Pi$ is randomly chosen, then π'' and P_π are identically distributed. Therefore, it is sufficient to show that π'' is one-way.

Similarly, to the proof of Claim 2.3.7, the permutation π'' has for every n enough many bits ($\log^2 n$) where it is a random permutation. Therefore, if the oracle machine makes $\leq n^c$ queries then the probability that M queries x is $\leq \frac{n^c}{2^{\log^2 n}}$. Here, the details are similar to the proof of Claim 2.3.7.

We now conclude the proof by applying the Borel-Cantelli lemma, similarly to the proof of Claim 2.3.7. $\square$

Chapter 3

An approach to randomness

This chapter evolves around the idea of parametrized access to the *auxiliary memory* in space-bounded computation. Our main focus is on auxiliary tapes containing random bits, where we parametrize on the number of passes over the random tape of a Stack Machine. Regardless of possible conceptual implications, this approach brings together technical tools developed in the areas of Computational Complexity for Stack Machines, Derandomization, Streaming, and Communication Complexity. The hope is to obtain technical insight through this new perspective. We make the first steps by introducing the model, we relate it to known questions in derandomization, and we study in detail variants of the model. Certain success in these variants serves as an indication that similar techniques possibly apply to the original Stack Machine model.

The main contribution of this part of the thesis is conceptual. Here, our exposition is devoted to the technical developments. The content of this chapter can be better appreciated if it is read together with the discussion on the concepts of Section 1.2. In Section 3.1 we provide necessary definitions and notation. The main model, logspace bounded Stack Machines, is used to define a hierarchy between P and RP. In Section 3.2 our main model is shown to have an alternative characterization, which is used to derive connections to previously known questions in derandomization. In Section 3.3 we show the derandomization of the first level of the two-sided error hierarchy. In the next two sections, that occupy a significant portion of the chapter, we study variants of the original model. In Section 3.4 we consider probabilistic variants of the Stack Machine model with certain success in the derandomization of a related hierarchy. In Section 3.5 we consider auxiliary tapes containing non-deterministic bits. Among others, in this last section we relate parametrized access to NP-witnesses with width parametrizations of SAT, which is also of independent interest. As a corollary we obtain an interesting and natural example of a language showing that under the Exponential Time Hypothesis, $\mathsf{PolyLogSpace} \not\subseteq \mathsf{P}$.

3.1 Definitions and preliminaries

The Complexity Theory definitions and conventions of this chapter are also relevant in chapters 4 and 5.

3.1.1 Notational conventions.

All logarithms are base 2. Typically $n \in \mathbb{Z}^+$ denotes the input length. U_n denotes the uniform distribution over $\{0,1\}^n$. For $x, y \in \{0,1\}^m$, $x \oplus y$ denotes the bitwise XOR of x and y. We use standard names for complexity classes, e.g. P, NP, BPP, NC, NEXP, $\mathsf{DSPACE}(f(n))$. Let $\mathsf{E} := \cup_{c>0}\mathsf{DTIME}(2^{cn})$, and $\mathsf{EXP} := \cup_{k=1}^{\infty}\mathsf{DTIME}(2^{n^k})$, $\mathsf{QuasiP} := \cup_{i=1}^{\infty}\mathsf{DTIME}(2^{\log^i n})$. For detailed definitions of Turing Machines (TMs), Alternating Turing Machines (ATMs), and space-bounded Stack Machines (previously named AuxPDAs) see e.g. [AB09, Coo71]. Similarly, for circuits and circuit classes cf. [AB09, Vol98]. We simplify notation by omitting ceilings for integer values.

3.1.2 Machine models, circuits, and conventions

Combinatorial circuits. We briefly recall at a high-level the definitions of relevant circuit classes. A boolean circuit C is a vertex-labeled directed acyclic graph with ordered vertices of three types: (i) *input gates* of fan-in 1, (ii) *gates* realizing the boolean functions $\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}$, and (iii) an *output gate* with fan-out 0. The edges of the circuits are called *wires*. For a given input $x \in \{0,1\}^n$ the output of the circuit is uniquely defined in the usual way, when realizing the usual boolean functions. The *size of* C, $|C|$, is defined to be the number of wires, and the *depth of* C, $\mathsf{depth}(C)$, is the longest path from any input to the output. For families of circuits, i.e. collections of circuits defined for every input length n or an infinite set of input lengths, we distinguish between (i) bounded fan-in (for the $\mathsf{AND}, \mathsf{OR}$ gates), (ii) unbounded fan-in, and (iii) semi-bounded fan-in circuit families. We also distinguish between (i) uniform and (ii) non-uniform circuits, depending on whether we insist that the circuits of a family can be computed by a Turing Machine. In this chapter all uniform circuit families are logspace uniform. NC^i, $\imath \geq 1$ is the class characterized by families of uniform, bounded fan-in, polysize circuits of depth $O(\log^i n)$, AC^i is the unbounded fan-in analog of NC^i, and SAC^i is defined similarly to AC^i, with all negations to the input level, where the AND gates have unbounded fan-in and the OR gates are of bounded fan-in. It easily follows that $\mathsf{AC}^{i-1} \subseteq \mathsf{NC}^i \subseteq \mathsf{SAC}^i \subseteq \mathsf{AC}^i$, $i \geq 1$.

We assume that the computation of Turing Machines is always halting, and every branch of the computation of non-deterministic machines is finite.

Alternating Turing Machines. An *Alternating Turing Machine* is a nondeterministic Turing Machine whose states are divided into two sets: existential states and universal states. An existential state is accepting if some transition leads to an accepting state; a universal state is accepting if every transition leads to an accepting state; and the computation is accepting if the initial state is accepting.

Stack Machines. In this chapter we consider two types of Stack Machines. A Stack Machine (AuxPDA) is a space bounded Turing Machine, equipped with a pushdown storage (or stack). For detailed definition of an SM and its computation see e.g. [Coo71]. We assume that all SMs halt with an empty stack. Furthermore, in each transition exactly one of the following happens:either a head-move, or a push to the stack or a pop from the stack. All Stack Machines are deterministic and they are logspace bounded unless mentioned otherwise. The first type of Stack Machines is that of a *Verifier Stack Machines* (vSMs). These are SMs with access to a read-only random (or nondeterministic) *external/auxiliary* tape; i.e. polytime verifiers (see Equation 3.1.1 below). The second type, is that of *non-uniform Stack Machines (nu-SMs)*, which are vSMs with an additional read-only polynomially long tape containing a non-uniform advice; note that we do not use the term *auxiliary tape* to refer to the advice tape. We *only* parametrize vSMs and nu-SMs with respect to the number of head reversals on the auxiliary random (or non-deterministic) tape. That is, for every other tape: work tape, input tape, non-uniform tape, the number of head reversals is not relevant. We avoid the use of the term *reversal* and instead we use the term *pass*; note that in logspace a head reversal can be simulated with a full scan of the auxiliary tape. This way we avoid confusion wrt the concept of "reversal complexity"[1].

For Stack Machines we define surface and full configurations (instantaneous descriptions) of their computation.

Definition 3.1.1 (Surface and Full configuration)**.** Let M be a SM computing on input x. The computation of $M(x)$ is defined in the usual way [Coo71] to be a sequence of valid configurations (full configurations). A *surface configuration* C of M on x consists of (1) the state of M, (2) the location of the input head; (3) the full configuration of the work tapes (head positions and tape contents); (4) the top stack symbol; and (5) the location of the external tape head (denoted by $\mathsf{head}(C)$) and the symbol it is scanning. A *full configuration* $\overline{C}$ of M on x consists of everything included in a surface configuration, plus (6) the entire stack content.

[1] This is a somehow esoteric issue for Computational Complexity, unrelated to this research which deals with a new concept.

Verifier Turing Machines. A *Verifier Turing Machine* (vTM) is a logspace bounded Turing Machine (i.e. without a stack) equipped with a polynomially long, read-only, two-way access auxiliary tape. We parametrize on the number of passes over the auxiliary tape.

3.1.3 Preliminaries for Stack Machines and vSMs

Stack Machines characterize[2] P. Furthermore, finer containments hold, in the sense that SMs with bounded time can be thought as polysize circuits of small depth. Let $\mathsf{SM}(t(n))$ be the class of languages decided by logspace SMs in time $O(t(n))$. Note that our SMs are deterministic, whereas in the literature it is more common to consider SMs that flip non-deterministic bits. By [BCD$^+$89, Coo71, Ruz81] we have the following inclusions.

$$\mathsf{NC}^1 \subseteq \mathsf{SM}(n^{O(1)}) \subseteq \mathsf{SAC}^1 \subseteq \mathsf{NC}^2 \subseteq \mathsf{SM}(2^{O(\log^2 n)}) \subseteq \mathsf{SAC}^2 \subseteq \ldots$$
$$\ldots \subseteq \mathsf{SAC} = \mathsf{NC} \subseteq \ldots \subseteq \mathsf{P} = \mathsf{SM}(2^{n^{O(1)}}) \qquad (3.1.1)$$

Recall that every probabilistic polytime TM computes a deterministic predicate $R(x, \rho)$, $\rho \in_R \{0,1\}^{n^k}$ where ρ is the random bits. vSMs compute arbitrary polytime predicates and an easy modification of the proofs in [Coo71] yields:

Fact 3.1.2. *A vSM with the standard definition of error characterizes the corresponding probabilistic polynomial time class such as $\mathsf{RP}, \mathsf{coRP}, \mathsf{BPP}, \mathsf{PP}$. Furthermore, if the external tape contains non-deterministic bits (equivalently: false-negatives with unbounded error) then we characterize NP.*

Also, we occasionally use the following, easy to see (e.g. [Ruz81]), fact.

Fact 3.1.3. *Let M be a $s(n)$ space-bounded SM, which halts on every input. Then, the maximum stack height during the computation of M is $2^{O(s(n))}$.*

3.1.4 Randomized and non-deterministic hierarchies

We consider one-sided false-negatives/false-positives unbounded/bounded error regimes corresponding in the usual way to the prefixes N-, coN-, R-, coR-, whereas are for two-sided unbounded/bounded error we use the prefixes P- and BP-. Furthermore, we consider bounding the number of passes $r(n)$ a vSM is allowed on its external tape. Recall that "PdL" stands for "push-down logspace". Accordingly, $x\mathsf{PdL}[r]$ is the class of languages accepted by vSMs that

[2]The stack is indispensable from the *general* study of probabilistic polynomial time classes, in the sense that logspace TMs without a stack characterize $\mathsf{LogSPACE} \subseteq \mathsf{NC}^2$. By adding a polynomially long (two-way access) random tape to logspace machines we define [Nis93b] the class $\mathsf{R}^*\mathsf{LogSpace} \subseteq \mathsf{RNC}^2$. Also, it is an easy exercise to see that using a queue instead of a stack we can compute every decidable set.

are allowed at most $r-1$ head reversals (i.e. r passes) on the external tape, operating in error regime x; we define $\mathsf{RPdL}[0]$ to be the class characterized by SMs without reading the random tape. For example, $\mathsf{RPdL}[1]$ is the class of languages accepted by vSMs with one-way access to the external tape and 1-sided error. We use the term *non-deterministic hierarchy* to refer to the collection of levels of $\mathsf{NPdL}[\cdot]$. Similarly, when the error condition is clear from the context we use the term *randomized hierarchy*.

It is easy to check that the following classes are closed under logspace (or weaker) transformations: $\mathsf{RPdL}[c]$, $c \in \mathbb{Z}^+$, $\mathsf{RPdL}[\mathsf{constant}] := \bigcup_{k=1}^{\infty} \mathsf{RPdL}[k]$, $\mathsf{RPdL}[O(\log n)] := \bigcup_{c>0} \mathsf{RPdL}[c \log n]$, and $\mathsf{RPdL}[\mathsf{poly}] := \bigcup_{c>0} \mathsf{RPdL}[n^c]$.

Remark 3.1.4. Currently we do not have any natural candidates for the lower levels of RPdL. This is a general issue for randomized polynomial time. With the exception of Polynomial Identity Testing (PIT) cf. [AB09], there are no other notable candidates for coRP or BPP. We can artificially construct such problems. Consider an arbitrary P-complete problem, say the circuit value problem CVP (cf. [AB09]), and PIT restricted to arithmetic circuits of depth $O(\log^i n)$; such restricted instances can be decided in coRNC^i. Then, concatenating yes instances of CVP and no instances of this restriction of PIT (of about the same length) we construct $L \in \mathsf{RPdL}[2^{O(\log^i n)}]$. Under the current state of knowledge we don't know whether $L \in \mathsf{P}$ or $L \in \mathsf{RNC}$. The reader should not be misled by this example. Languages in $\mathsf{RPdL}[2^{O(\log^i n)}]$ are much more general than a computation consisting of a P and a separate RNC phase (see Section 3.1.5).

3.1.5 P+RNC^i: polytime Randomness Compilers

$\mathsf{P}+\mathsf{RNC}^i$ denotes the class of languages such that for every $L \in \mathsf{P}+\mathsf{RNC}^i$: there exists a polynomial time TM M which on every input x, M computes a probabilistic circuit C_x of constant fan-in, polynomial size, and depth $O(\log^i n)$. If $x \in L$, then $\Pr_\rho[C_x(\rho) = 1] \geq \frac{1}{2}$, whereas if $x \notin L$ then $\Pr_\rho[C_x(\rho) = 1] = 0$. We refer to this as the *P+RNC model*, and we call the polytime transducer M a *Randomness Compiler*. Similarly, we define $\mathsf{P}+\mathsf{BPNC}$.

3.1.6 Variants of the model, path-width and SAT

There are additional definitions regarding variants of the model, parametrizations of subsets of SAT, and relevant graph theoretic definitions and terminology. For *pathwidth* the definitions become relevant also for Chapter 5. We give these definitions in the relevant sections 3.3 and 3.4.

3.2 RPdL$[n^{\mathsf{polylog}n}]$ and P+RNC

This section contains the main results for the RPdL hierarchy as discussed in Section 1.2. Everything mentioned here holds also for the 2-sided error analog, BPPdL. We have introduced two models to define restricted use of randomness. The first model (characterization of RPdL) is a transition system on which we make explicit the concept of essential use of randomness, by counting random bits accesses. In the second model (characterization of P+RNC) we compile all the needed randomness in a shallow circuit. Here is an intuitive way to think about these views.

- A randomness compiler provides an easier way to think about an algorithm; i.e. when placing problems in the hierarchy.
- The vSM view provides a possibly easier way to obtain derandomization; compare to Section 3.4.1.

A key idea in relating RPdL with P+RNC is the use of the time compression Lemma 3.2.1. Recall that there are (logspace) Stack Machines that work in time exponential in the input length. This compression becomes possible when we have an additional (non-uniform or poly-time computable) advice. Our proof is based on the fact that the computation between two successive head moves on the input can be compressed to be polynomially-long, given some small (polysize) advice which depends on the input length. At first, the feasibility of this "compression" may be seen as counter-intuitive since the computation between two successive head-moves depends on the content of the stack, which in turn it depends on the given input (not only its length). In a somewhat different context and through a different argument Allender [All89] independently obtains a similar result.

Lemma 3.2.1 (Time-compression lemma)**.** *Let M be a deterministic nu-SM which makes $r(n)$ passes over its input. Then, there exists a deterministic nu-SM M' which makes $r(n)$ passes over its input, it works in time $O(r(n)n^{O(1)})$, and decides the same language as M.*

Proof. It suffices to show that the computation between two successive input head-moves can be "compressed" to be polynomially long by the use of a non-uniform advice. This non-uniform advice extends the non-uniform advice already given to the machine. Fix two arbitrary successive head-moves. Partition the computation γ between these head-moves in two phases. Suppose that immediately after reading the input symbol the stack level is at l. In phase 1, γ reaches its lowest stack height l_{min}. Let γ_1 be the computation subsequence of γ from the beginning until we reach the lowest stack level and just before we start going upwards (pushing symbols to the stack). Define γ_2 to be the complement of γ_1 wrt γ. Hence, in γ_2

the computation reaches its final stack height l_{last}. By Fact 3.1.3 the stack height of M is polynomial, and therefore the stack height in γ_1 gets decreased at most polynomially lower (from l to l_{min}) and in γ_2 gets increased polynomially higher (from l_{min} to l_{last}). We construct M' simulating M using the following non-uniform advice. The advice is a function from every surface configuration to the set of surface configuration together with two special symbols $\{\uparrow, \downarrow\}$. For every surface configuration σ define exactly one of the three pairs:

1. If starting from σ we can return to the same stack level without ever going below the initial stack-level (and without a head-move on the input) then consider the configuration after a maximally-long computation such that when M returns to the same stack-level the surface configuration is σ'. Then, the corresponding pair is (σ, σ').

2. If starting from σ we move at least one level upwards without ever returning to the initial stack-level (and without moving the head) then the pair is $(\sigma, \uparrow)$.

3. Else, the pair is $(\sigma, \downarrow)$.

Obviously, this is a well-defined function and we say that a surface configuration σ is of type (1), (2) or (3) respectively.

Between two successive head-moves M' simulates M by reading the advice tape and updating its surface configuration appropriately. In case of (1) it updates the worktape, the state and the top stack symbol. In case of (2) and (3) it simulates M for one step.

We refer to a *simulation step* as the computation sequence of M' in which M' reads the non-uniform tape, compares it to the current surface configuration and updates the surface configuration appropriately. In what follows the reader is reminded that γ_1, γ_2 is the computation of M which is simulated by the machine M', and that M' is given the non-uniform advice. We say that a function from the integers is 2-monotonically increasing (decreasing) if it is strictly increasing (decreasing) for two successive integers; i.e. for the function $h : \mathbb{Z}^+ \to \mathbb{Z}^+$, $h(n) \leq h(n+1)$ and $h(n) < h(n+2)$.

Claim 3.2.2. *In the simulation of γ_1 the stack height in M' is 2-monotonically decreasing. Hence, this simulation takes at most $2(l - l_{min})$ simulation steps of M'.*

Proof. Consider two successive stack levels $l_1 > l_2 := l_1 - 1$ in γ_1 and consider the first time M' gets to l_1. The current surface configuration σ_1 cannot be of type (2). Suppose that σ_1 is of type (2). Since we are in γ_1 we know that the stack level gets as low as l_{min}. If σ_1 is of type (2) then we know that during γ_1 the stack level will get back to l_1. Hence, σ_1 should instead be of type (1).

Hence, σ_1 is either of type (1) or of type (3). If it is of type (3) there is nothing left to show. Suppose σ_1 is of type (1). The fact that the next surface configuration in the simulation cannot be of the same type (1) follows by the maximality in the definition of type (1). □

Similarly, we show that in the simulation of γ_2 the stack height in M' is 2-monotonically increasing. □

Let us repeat the statements and prove the theorems which appear in Section 1.2.

Theorem 3.2.3. *Let $i \in \mathbb{Z}^+$. Then, $\mathsf{P}\text{+}\mathsf{RNC}^i \subseteq \mathsf{RPdL}[2^{O(\log^i n)}] \subseteq \mathsf{P}\text{+}\mathsf{RNC}^{i+1}$.*

Proof. $\mathsf{P}\text{+}\mathsf{RNC}^i \subseteq \mathsf{RPdL}[2^{O(\log^i n)}]$ is the easy inclusion. Let $L \in \mathsf{P}\text{+}\mathsf{RNC}^i$ and M be the polytime transducer that on input x computes C_x. Construct a vSM M' that on input $\langle 1^{|x|}, k\rangle$ works as follows: use Cook's algorithm [Coo71] to simulate M so as to obtain the k-th output bit from the description of C_x. Note that each bit is computed without accessing the random tape. M' evaluates C_x on the provided randomness in the usual way (e.g. [Ruz81]) by a depth first search. Note that if the description of the circuit were given through oracle access, the evaluation procedure would have taken time $2^{O(\log^i n)}$. Hence, M' makes at most $2^{O(\log^i n)}$ reversals on the random tape and the accepting probability is the same as that of C_x.

To show $\mathsf{RPdL}[2^{O(\log^i n)}] \subseteq \mathsf{P}\text{+}\mathsf{RNC}^{i+1}$ we rely on the time-compression Lemma 3.2.1 and on [Ruz80, Ruz81]. In the version of the time-compression lemma for uniform machines, the advice can be computed in polynomial time.

Let M' be a vSM witnessing $L \in \mathsf{RPdL}[2^{O(\log^i n)}]$, for an arbitrary such L. We will show how to construct a polytime M that on input x outputs a circuit C_x with the same accepting probability as M'.

There exists M'' extending M' as follows: M'' is M' as described in the proof of Lemma 3.2.1; i.e. it has an additional read-only advice tape and it works as specified in the proof of Lemma 3.2.1. Therefore, on input x given that the extra-tape contains this advice, M'' computes identically to M'. Syntactically, M'' is a SM with three read-only input tapes. When the 3rd tape contains the appropriate advice, M'' is a SM that works in space $O(\log n)$ and in time $2^{O(\log^i n)} n^{O(1)} = 2^{O(\log^i n)}$, $i \geq 1$. We assert the existence of an equivalent ATM M_{ATM} that works in space $O(\log n)$ and in time $O(\log^{i+1} n)$. It is straightforward to verify that all equivalences between SMs and ATMs in the constructions of Theorem 5 part 3 [Ruz81] p.379 (i.e. Theorem 2 [Ruz80] pp. 227-231), and Corollary 3 (c,d,e) [Ruz81] pp. 379-380 are the same when instead of one we have three inputs tapes. Hence, syntactically given the 3-input tape SM M'' we have an ATM M_{ATM} with 3-input tapes that computes identically. The constant description of M_{ATM} can be hardwired in a (polytime) TM M. Although M_{ATM} and M'' accept the same inputs, we are only interested in the computations where their 3rd tape contains the

advice of Lemma 3.2.1; in which case M'' computes the same as M'. Intuitively, one can blur the distinction between space-time bounded ATMs and size-depth families of combinatorial circuits, and moreover we observe that given the description of the ATM we can construct efficiently the circuit for the corresponding input length. That is, the description of the polytime M should be evident through the observation that the construction in the proof of Theorem 3 [Ruz81] p.375 is computable in time $2^{O(S(n))}O(T(n))n^{O(1)} = n^{O(1)}$, where $S(n) = O(\log n)$ and $T(n) = O(\log^{i+1} n)$ is the space and the time of M_{ATM}. For completeness we briefly review this construction below.

1. On input x use (the modified) Cook's algorithm to compute the advice of Lemma 3.2.1, which is a function of $n = |x|$.

2. M has hardwired the description of the ATM M_{ATM} and it computes the description of a circuit C_x. In this circuit, both the input x and the advice are hardwired using the constant 0/1 gates of the circuit.

3. The circuit gates are labelled with (α, t), where α is the configuration of the ATM, and t is the time, where the output gates has label $(\alpha_{\text{initial}}, 0)$, where α_{initial} is the starting configuration. Configurations of type $\forall, \exists$ correspond to gates $\wedge, \vee$, we connect gates (α, t), $(\beta, t+1)$ if α yields β. The only exceptions to this rule is when the time and the space becomes bigger than $T(n), S(n)$ in which case we hardwire the gates to 0, and when we have configurations accessing the input in which case instead of a gate we have an input gate.

Step (1) takes polynomial time. The construction of the circuit in Step 3, also takes polynomial time $(2^{O(S(n))}O(T(n))n^{O(1)})$. □

Combining Theorem 3.2.3 and Lemma 3.2.1 we can derandomize $\mathsf{RPdL}[2^{\log^{O(1)} n}]$ using PRGs that derandomize RNC. In fact, we present the stronger argument for BPPdL since the actual constants in the proof matter. For this we rely on PRGs introduced in Definition 1.2.3. We remark that our PRGs definitions are more qualitative than usual. Several parameters have been fixed to reduce clutter. For example, we focus on PRGs that stretch polylogarithmic bits to polynomial. These parameters can be adjusted in the standard way to generalize our results. In what follows, all distinguishers are non-uniform circuits.

Theorem 3.2.4. . *Let $k, i \geq 1$. Suppose that there exists a $(\mathsf{NC}^{i+1}, k, \frac{1}{7})$-PRG. Then, $\mathsf{BPPdL}[2^{O(\log^i n)}] \subseteq \mathsf{DTIME}(2^{O(\log^k n)})$.*

Proof. These types of arguments are standard in derandomization. Observe that the proof of Theorem 3.2.3 carries through in case of two-sided error. Therefore, we have the following.

Theorem. *Let $i \in \mathbb{Z}^+$. Then, $\mathsf{P}+\mathsf{BPNC}^i \subseteq \mathsf{BPPdL}[2^{O(\log^i n)}] \subseteq \mathsf{P}+\mathsf{BPNC}^{i+1}$.*

Let $L \in \mathsf{BPPdL}[2^{O(\log^i n)}]$. Then, there exists polytime transducer M that on input x outputs a circuit C_x of depth $O(\log^{i+1} n)$, such that if $x \in L$ then $\Pr_\rho[C_x(\rho) = 1] \geq 2/3$, whereas if $x \notin L$ then $\Pr_\rho[C_x(\rho) = 1] \leq 1/3$, and the input to the circuit is of length n^m, for a constant $m > 0$. We construct a deterministic TM $\hat{M}$ that runs in time $O(2^{\log^k n})$. Here is the description of $\hat{M}$: (1) Enumerate all strings of length $O(m^k \log^k n)$. (2) For each such string use the PRG to compute a string ρ of length n^m. (3) Fix the random tape of M to be ρ and simulate M in polytime. (4) Output the majority of the outcome of the simulation.

$\hat{M}$ takes time $2^{O(\log^k n)}$ to enumerate all strings, for each such string it takes polynomial time to compute ρ and then polynomial time to simulate M on (x, ρ).

We claim that for all large x_i's:

> (*) for input x_i, at least a fraction $\frac{2}{3} - \frac{1}{7} = \frac{11}{21} > \frac{1}{2}$ of the pseudorandom strings cause M to give the correct answer.

Note that the error probability in the definition of $\mathsf{P}+\mathsf{BPPdL}$ can be amplified to any e.g. constant arbitrarily close to 0, and thus (*) suffices to conclude the theorem.

Suppose (*) is not true. Then, we are going to use M to construct a (non-uniform) family of distinguishers D_n, for infinitely many n's (where n takes values among the output length of the PRG). Let $\{x_i\}$ be an infinite family of inputs where (*) is false. Then, by definition C_{x_i} is an appropriate distinguisher with distinguishing probability strictly greater than $\frac{2}{3} - \frac{11}{21} = \frac{1}{7}$. □

Note that the hypothesis in Theorem 3.2.4 involves an assumption against non-uniform adversaries. Most such unconditional proofs (e.g. [Nis92, INW94]) are against non-uniform adversaries. A minor modification in Cook's construction [Coo71] shows that given the additional advice for M (or if M is uniform), then the non-uniform additional advice in the proof of Lemma 3.2.1 can be replaced with a polytime computable advice (as used in the proof of Lemma 3.2.4). It turns out that one cannot do better than this.

Lemma 3.2.5. *Given the advice for M, there does not exist M' which achieves the time-compression of Lemma 3.2.1 when the additional advice is computed in logspace uniform NC, unless $\mathsf{PSPACE} = \mathsf{EXP}$.*

Proof. This is easy considering the results in [All89]. In this proof all circuit classes are logspace uniform. Suppose that the additional (non-uniform) advice can be computed in $\mathsf{NC} = \mathsf{SAC}$; i.e. let $\hat{M}$ be an SM that computes an advice string (i.e. the i-th bit of the output) in NC; for example, say that this advice is as in the proof of Lemma 3.2.1. We show that every polytime computable tally language $L \subseteq \{0\}^*$ can be computed in NC. By [All89] (Corollary 6.3 and Corollary 6.7) this implies that $\mathsf{PSPACE} = \mathsf{EXP}$.

Since, $\mathcal{L}(\hat{M}) \in \mathsf{NC}$, by Equation 3.1.1 we know that M works in time $O(2^{\log^{O(1)} n})$.

Fix arbitrary $L \subseteq \{0\}^*$, $L \in \mathsf{P}$. Let M be a SM, such that $\mathcal{L}(M) = L$.

We construct a SM M'', which works in time $O(2^{\log^{O(1)} n})$, such that $\mathcal{L}(M'') = L$; i.e. $L \in \mathsf{NC}$.

Here is the description of M''. In a single tape-scan over 0^n compute n and store it on the work-tape. Note that M'' can simulate M on the given input, without moving its input head. Since the advice can be computed by $\hat{M}$ we can construct a uniform M' that decides L and it makes polynomial many calls to an oracle computing the advice bit given its index. By assumption each oracle call can be computed by simulating $\hat{M}$ in time $O(2^{\log^{O(1)} n})$. Therefore, M'' works in time $O(n^{O(1)} 2^{\log^{O(1)} n}) = O(2^{\log^{O(1)} n})$. □

3.3 Derandomization for 1-pass, 2-sided error: $\mathsf{PPdL}[1] = \mathsf{P}$

Theorem 1 in [Coo71] implies that $\mathsf{NPdL}[1] = \mathsf{P}$, and thus $\mathsf{RPdL}[1] = \mathsf{P}$, and $\mathsf{coRPdL}[1] = \mathsf{P}$ (P is closed under complement). However, there is no immediate implication for 2-sided error.

We present an algorithm[3] derandomizing the first level of the randomized hierarchy with 2-sided and unbounded error.

Theorem 3.3.1. *$\mathsf{PPdL}[1] \subseteq \mathsf{P}$.*

An immediate corollary is $\mathsf{BPPdL}[1] = \mathsf{PPdL}[1] = \mathsf{P}$.

3.3.1 Outline of the algorithm - comparison with [Coo71]

We look at the computation of a vSM M on an input x as a tree of depth $2^{n^{O(1)}}$ of configurations, where every branching node corresponds to tossing a coin (we can think of a one-way pass over the randomness vSM, as a SM that tosses coins). In the *nondeterministic (false-negatives unbounded)* error regime considered in [Coo71], to decide whether M accepts x, one has to determine whether there is a path from the initial to the accepting configuration.

Recall that the difference of a surface configuration from a regular configuration is that in the surface configuration we do not include the full stack content (just the top stack symbol).

Cook defines the notion of "realizable pair of surface configurations" to be a pair (C_1, C_2) such that there exists a computation path that takes M from C_1 to C_2, with the stack level being the same in both, and never going below that level between them. The key in Cook's proof is showing how existing realizable pairs can be combined (using a tabular/dynamic programming method) to yield new ones, until, eventually, all realizable pairs are found. The number of

[3] The main idea for the algorithm is due to Matei David. From the results in [Ven06] we can obtain a more complicated algorithm that runs in time $n^{O(\log n)}$.

distinct surface configurations is polynomial and thus this takes polytime. At the end, to decide whether M accepts x, we check if the surface of the initial configuration and that of the accepting configuration are realizable.

In the *two-sided unbounded* error regime considered in here, one way to decide if M accepts x is to compute the total probability of all strings that take M from the initial configuration to the accepting one. Recall that the external string (i.e. the content of the auxiliary tape) is of polynomial length and we include the head position in the surface configuration. Then, we only have to *count* the number of strings leading to the accepting configuration (since the auxiliary tape is polynomially long). Not unexpectedly, we are only able to perform this counting when M tosses at most polynomially many coins, in contrast to the nondeterministic regime [Coo71] where the proof works for exponentially many tosses. We consider pairs of surface configurations (C_1, C_2), and we now *count exactly* the number $\alpha(C_1, C_2)$ of strings that take M from C_1 to C_2, with stack level the same, and never going below that level in-between them. As compared to Cook, the rule for computing the values $\alpha(\cdot, \cdot)$ for new pairs from previously computed ones is more involved.

3.3.2 The algorithm and its proof of correctness

Let M be a vSM with one-way access to its external tape and let w be a string. We assume, without loss of generality, that M always halts with an empty stack, scanning the last symbol on the external tape. By definition of the P- (two-sided unbounded) error regime, M accepts w iff the number of random strings that take M to an accepting configuration is greater than the number of random strings that take it to a rejecting configuration. In what follows, we show that there is a poly-time algorithm that, given M and w, outputs the exact number of random strings that take M to an accepting configuration. Clearly, the existence of such an algorithm places $L(M)$ in P.

We make the convention that every transition of M is of exactly one of the following types: a *push* transition, a *pop* transition, a (coin) *toss* transition (in which the external tape head moves right), and a *move* transition (in which any of the input tape head and internal tape heads can move).

Let $\overline{C_1}, \overline{C_2}$ be two full configurations of M on w with $\mathsf{head}(\overline{C_1}) \leq \mathsf{head}(\overline{C_2})$, and let ρ be a string of size $\mathsf{head}(\overline{C_2}) - \mathsf{head}(\overline{C_1})$ (when $\mathsf{head}(C_1) = \mathsf{head}(C_2)$, ρ is the empty string). We say that $(\overline{C_1}, \overline{C_2})$ *is up-realizable along* ρ if, when we plug in ρ on the external tape of M between locations $\mathsf{head}(\overline{C_1}) + 1$ and $\mathsf{head}(\overline{C_2})$, and we start M in full configuration $\overline{C_1}$, M eventually reaches full configuration $\overline{C_2}$, and the stack level never drops below the level in $\overline{C_1}$. We say that $(\overline{C_1}, \overline{C_2})$ *is realizable along* ρ if, additionally, the stack level is the same in $\overline{C_1}$ and in $\overline{C_2}$.

Informally, the heart of the proof in [Coo71], showing that a nondeterministic SM can be

simulated in polynomial time, is that one can compute all realizable pairs of surface configurations, where a pair is realizable (in their sense) if it is realizable (in our sense) along *some* string (in that terminology, a sequence of nondeterministic guesses). Here, to decide whether M accepts w, we compute exactly the number of strings along which each pair is realizable.

More formally, let $(\overline{C_1}, \overline{C_2})$ be a realizable pair of full configurations. We define $\alpha(\overline{C_1}, \overline{C_2})$ to be the number of random strings ρ, such that $(\overline{C_1}, \overline{C_2})$ is realizable along ρ. Crucially, observe that since $(\overline{C_1}, \overline{C_2})$ is realizable, the computation path that starts at $\overline{C_1}$ and ends at $\overline{C_2}$ only depends on the surface configurations C_1, C_2 respectively. In other words, if $(\overline{C_1'}, \overline{C_2'})$ is a pair of realizable pair of full configurations with surface configurations C_1 and C_2 (i.e. the same as for $\overline{C_1}, \overline{C_2}$), then $\alpha(\overline{C_1'}, \overline{C_2'}) = \alpha(\overline{C_1}, \overline{C_2})$. In this sense the notation $\alpha(C_1, C_2)$ is well-defined. Below, we give an algorithm computing all non-zero entries $\alpha(C_1, C_2)$. Having done that, to decide whether M accepts w, we simply sum the entries $\alpha(C_0, C_a)$, where C_0 is the surface of the initial configuration, and C_a ranges over the surfaces of accept configurations, and compare the resulting value with half times the maximum number of random strings.

Let C be a surface configuration of M. Note that C completely determines the next transition of M. If C is followed by a *move* or a *push* transition, let $\mathsf{next}(C)$ denote the next surface configuration of M (observe that, in this case, C completely determines $\mathsf{next}(C)$). If C is followed by a *toss* transition, slightly overloading notation, let $\mathsf{next}(C)$ denote the set consisting of the two possible surface configurations following C, corresponding to the two possible outcomes of the coin toss (i.e., of the next symbol read on the external tape). If C is followed by a *pop* transition, let $\mathsf{next}(C, x)$ denote the surface configuration following C in which the top stack symbol is now x. It is easy to verify the following consequences of our definitions.

Lemma 3.3.2. *For all surface configurations C_1, C_2:*

- *If $C_1 = C_2$, then $\alpha(C_1, C_2) = 1$; namely, only the empty string.*
- *If $C_1 \neq C_2$ is followed by a* move *transition, then $\alpha(C_1, C_2) = \alpha(\mathsf{next}(C_1), C_2)$, where next is defined as above.*
- *If $C_1 \neq C_2$ is followed by a* toss *transition, and $\mathsf{next}(C_1) = \{C_3, C_4\}$, then $\alpha(C_1, C_2) = \alpha(C_3, C_2) + \alpha(C_4, C_2)$.*
- *If $C_1 \neq C_2$ is followed by a* push *transition, denoting by x the top stack symbol in C_1,*

$$\alpha(C_1, C_2) = \sum_{C_3 : C_3 \text{ is followed by a pop transition}} \alpha(\mathsf{next}(C_1), C_3) \cdot \alpha(\mathsf{next}(C_3, x), C_2).$$

We now show how to compute the entries $\alpha(C_1, C_2)$. To that end, we first define a partial order relation on surface configurations, as follows. Informally, $C_1 \prec C_1'$ when some entry of

the form $\alpha(C_1, \cdot)$ directly depends on another entry $\alpha(C_1', \cdot)$. Formally, we write $C_1 \prec C_1'$ if (i) C_1 is followed by a *move* transition and $C_1' = \mathsf{next}(C_1)$; (ii) C_1 is followed by a *toss* transition and $C_1' \in \mathsf{next}(C_1)$; and (iii) C_1 is followed by a *push* transition and (iii-a) $C_1' = \mathsf{next}(C_1)$, *or* (iii-b) there exists a surface configuration C_1'' such that $\alpha(\mathsf{next}(C_1), C_1'') > 0$, C_1'' is followed by a *pop* transition, and $C_1' = \mathsf{next}(C_1'', x)$, where x is the top stack symbol in C_1.

Lemma 3.3.3. *The relation $\prec$ contains no cycles.*

Proof. Assume a cycle exists. Let $C_1, \ldots, C_m$ be a sequence of surface configurations such that: $C_1 = C_m$, and for every $1 \leq i < m$, $C_i \prec C_{i+1}$. Observe that whenever $C_i \prec C_{i+1}$, we have $\mathsf{head}(C_i) \leq \mathsf{head}(C_{i+1})$. Hence, $\mathsf{head}(C_1) = \mathsf{head}(C_m) = \mathsf{head}(C_i)$ for all $i \in [m]$, and so none of the C_i for $i < m$ is followed by a *toss* transition. Furthermore, if C_i is followed by a *pop* transition, we cannot have $C_i \prec C_{i+1}$ because such configurations create no dependencies. Hence, every one of the configurations in the sequence $C_1, \ldots, C_m$ is followed by either a *move* or a *push* transition.

But then, there is a path from a full configuration $\overline{C_1}$ to a full configuration $\overline{C_1}'$ with the same surface $C_1 = C_m$, with the stack level never dropping below that in $\overline{C_1}$. Since this path only depends on the surface C_1 of $\overline{C_1}$, and since $\overline{C_1}'$ has the same surface, this path is infinite. We assumed C_1 is reachable from the initial configuration of M, hence M does not halt along this path, a contradiction. □

To fill in the values in $\alpha(\cdot, \cdot)$, we proceed as follows.

1. for all surface configs C, $\texttt{seen}[C] \leftarrow \texttt{false}$
2. find any surface config C_1 such that $\texttt{seen}[C_1']$ for all C_1' with $C_1 \prec C_1'$
3. for all surface configs C_2
4. compute $\alpha(C_1, C_2)$ using the formulas in Lemma 3.3.2
5. endfor
6. $\texttt{seen}[C_1] \leftarrow \texttt{true}$
7. repeat step 2 as long there are unseen surface configurations

The test in step 2 can easily be implemented in the obvious way suggested by the definition of $\prec$. In the "most complex" case (iii-b), one can simply try out all possibilities for C_1'', since there are only polynomially many of them. By Lemma 3.3.3, this algorithm does not get stuck at step 2. As an invariant, when line 6 is executed, all non-zero entries in the row $\alpha(C_1, \cdot)$ have been computed. Since there are polynomially many surface configs, the algorithm runs in polynomial time.

3.4 Variants of the model: probabilistic variants and the fooling power of Nisan's PRG against multiple passes

We consider variants of probabilistic vSMs. In our treatment, the most interesting one is that of a probabilistic vTM; i.e. a logspace Turing Machine equipped with a two-way, polynomially long random tape. We study probabilistic vTMs and consider the possibility of derandomizing their computation by studying the resilience of the classical Nisan's PRG [Nis92] against *multiple* passes. This PRG is shown to fool logspace machines that make a single pass over the tape that contains the random or the pseudorandom string.

This research direction *conceptually* relates to the derandomization of RPdL and BPPdL. The hope in studying the resilience of PRGs that fool Stack Machines with a single pass over their randomness, is that we may be able to obtain results similar to those for vTMs. This seems to be a feasible goal. Under Theorem 3.2.3 we need PRGs that fool at least BPNC^2 circuits to obtain derandomization of any level of BPPdL. However, the derandomization of the first few levels of BPPdL and RPdL may be easier, and in particular may be possible without the full derandomization of BPNC^2 or even BPNC^1.

In Section 3.4.1 we introduce and partially derandomize a hierarchy[4] much lower than P and in particular lower than BPPdL. We consider logspace machines without a stack, and we parametrize on the number of passes over the random tape. This defines a hierarchy of classes between $\mathsf{LogSpace}$ and $\mathsf{BP}^*\mathsf{LogSpace} \supseteq \mathsf{BPNC}^1$ [Nis93b]. Therefore a full derandomization of this hierarchy derandomizes BPNC^1. Independent to a partial derandomization of BPPdL, the derandomization of BPNC^1 is a major open question by itself.

3.4.1 Nisan's PRG and the derandomization of BPNC^1

We study the relation between the derandomization of BPNC^1 and the fooling power of the classical space-bounded PRG of Nisan [Nis92].

Observe that a PRG that fools logspace machines with two-way access over the input, also fools NC^1 circuits. The classical PRG of Nisan [Nis92] is shown to fool logspace machines that make $r(n) = 1$ pass over the random or pseudorandom input. Using a simple analysis, we show that for every $k > 0$ there exists k' such that for seeds of length $\log^{k'} n$ the original Nisan's PRG is also secure against logspace machines with $r(n) = \log^k n$ passes. This result uses Nisan's PRG in a black-box fashion, and holds for a broad class of PRGs. Prior to our work, Impagliazzo, Nisan and Wigderson [INW94] gave a different (and more complicated) PRG that works against a small number of passes; in fact, their result holds in a somewhat more general

[4]For this variant we consider 2-sided error classes, since for 1-sided error, part of our results can be derived by applying lemmas from Section 3.5.2.

setting. Though, this property of the Nisan's original PRG was previously unknown.

We complement the above result by showing that in general, Nisan's PRG cannot fool non-uniform *logspace* machines with $r(n) = n^{O(1)}$ unidirectional passes[5] over the input, even with a *seed of length* $2^{c\sqrt{\log n}}$, for any $c > 0$. Our adversary uses some elementary algebraic properties. It relies on a deep result due to Mulmuley [Mul87] (which derandomizes Borodin, Gathen and Hopcroft [BGH82]) for solving a non-singular system of linear equations with poly-size circuits of $O(\log^2(n))$ depth, which in particular implies a $O(\log^2 n)$ space algorithm. Our adversary deals with linear systems of size only $O(2^{c\sqrt{\log n}})$, and we can therefore achieve space $O(\log n)$; note that the whole input is of length n.

This approach falls into a more general research direction. To date there are explicit constructions of PRGs that fool (non-uniform) logspace distinguishers with a single pass over the input. We ask what are the limitations of these PRGs when there are multiple passes.

Definition 3.4.1 (Passes-space resilient PRG)**.** Let $G : \{0,1\}^m \to \{0,1\}^n$, $m < n$, be a function computable in time polynomial in n. We say that $G : \{0,1\}^m \to \{0,1\}^n$ is a *pseudorandom generator against space* $S(n)$ *and passes* $r(n)$ *with parameter* $\varepsilon(n)$, if for every S-space bounded Turing Machine Adv that makes at most r passes (left-to-right) over its input and sufficiently large n we have

$$\left| \Pr[\mathsf{Adv}(\rho) \text{ accepts}] - \Pr[\mathsf{Adv}(G(\rho')) \text{ accepts}] \right| \leq \varepsilon$$

where $\rho \in_R \{0,1\}^n$, and $\rho' \in_R \{0,1\}^m$.

A hierarchy between LogSpace and BP*LogSpace. Nisan [Nis93b] defines as $\mathsf{BP^*LogSpace}$ the class of languages decided by logspace machines with an auxiliary read-only, polynomially long, and two-way tape. In particular, $\mathsf{BPNC^1} \subseteq \mathsf{BP^*LogSpace}$. We define $\mathsf{BPL}[r(n)]$ to be the class of languages decided by logspace machines with at most $r(n)$ passes over the randomness, and 2-sided error bounded away from 1/2. Thus,

$$\mathsf{BPNC^1} \subseteq \mathsf{BP^*LogSpace} = \mathsf{BPL}[n^{O(1)}] := \cup_{k>0}\mathsf{BPL}[n^k]$$

We also know that $\mathsf{NC^1} \subseteq \mathsf{LogSpace} = \mathsf{BPL}[0]$. In this sense, derandomizing along $\mathsf{BPL}[r(n)]$ indicates progress towards the derandomization of $\mathsf{BPNC^1}$. Also, let $\mathsf{BPL}[\mathsf{polylog}] := \cup_{k>0}\mathsf{BPL}[\log^k n]$.

Notation. Let $G : A \to B$ be an arbitrary function, and let D be a probability distribution on A. We denote by $G(D)$ the probability distribution of the random variable $G(x)$, where x is chosen from D.

[5] This is the only place in the thesis where "passes" are defined differently. In every other place r passes mean $r-1$ head reversals. Here the term "pass" denotes a left to right scan of the tape.

Let $N \geq 1$, and let D be a probability distribution over $\{0,1\}^N$. For any $t \geq 1$, we denote by D^t the probability distribution over $\{0,1\}^{Nt}$ obtained by taking t concatenated copies of a string x that is chosen from D.

Let D_1, D_2 be probability distributions over a finite set A. We use the notation

$$\|D_1 - D_2\|_1 = \sum_{x \in A} \left| \Pr_{y_1 \in D_1}[y_1 = x] - \Pr_{y_2 \in D_2}[y_2 = x] \right|.$$

A non-uniform, one-way TM M is equivalent to a family of *Finite State Machines* (FSMs). Let Q be a FSM with alphabet $\{0,1\}$, and with a designated initial state, and let D be a probability distribution over $\{0,1\}^n$. Then, we denote by $Q(D)$ the matrix whose (i,j)-th entry is the probability getting from state i to state j after reading a string x chosen from D.

For a matrix $P \in \mathbb{R}^{N \times N}$ let $\|P\|_1 = \max_i \sum_j |P_{i,j}|$.

Derandomizing the first polylog levels: BPL[polylog] $\subseteq$ QuasiP

Since we deal with space-bounded machines realized as FSMs, the following definition is stronger than Definition 3.4.1.

Definition 3.4.2 (Pseudorandom generator against FSMs)**.** Let $G : \{0,1\}^m \to \{0,1\}^n$ be computable in time polynomial in n, and let $\varepsilon, w > 0$. We say that G is a *pseudorandom generator against FSMs* for space w with parameter ε if for any FSM Q with 2^w states, we have $\|Q(U_n) - Q(G(U_m))\|_1 \leq \varepsilon$.

Theorem 3.4.3 (Nisan [Nis92])**.** *There exists $c > 0$ such that for any $N > 0$, there exists a function $G : \{0,1\}^{N^2} \to \{0,1\}^{N 2^{cN}}$ computable in time polynomial in 2^{cN} which is a pseudorandom generator against FSMs for space cN, with parameter 2^{-cN}.*

Instead of considering an adversary as an FSM with multiple passes over the input, it is convenient to consider a one-way FSM whose input contains multiple copies of the original input. For a function $G : \{0,1\}^m \to \{0,1\}^n$ we define $G^2 : \{0,1\}^m \to \{0,1\}^{2n}$, $x \mapsto G(x) \circ G(x)$, where $\circ$ denotes concatenation of strings. The following simple lemma states that Nisan's PRG works (with a small loss in the parameter) even if the space-bounded machine is allowed to make two passes over the random tape.

Lemma 3.4.4. *Let $G : \{0,1\}^m \to \{0,1\}^n$ be a PRG against FSMs for space $2s$ with parameter ε. Then $G^2 : \{0,1\}^m \to \{0,1\}^{2n}$ is a PRG against FSMs with space s with parameter $\varepsilon' = \varepsilon \cdot 2^{2s}$.*

Proof. Suppose that G^2 is not pseudorandom against FSMs of space s with parameter ε'. We show that G is not a PRG against FSMs for space $2s$ with parameter ε.

Therefore, there exists a FSM with state space $\{1, \ldots, 2^s\}$ such that

$$\|Q(G^2) - Q(U_n^2)\|_1 > \varepsilon' \tag{3.4.1}$$

Let 1 be the initial state of Q. For a string $x \in \{0,1\}^n$, and for $i, j \in \{1, \ldots, 2^s\}$ we write $i \xrightarrow{x}_Q j$ to denote the event that starting from state i and reading the string x, the FSM Q ends up to state j. For any $i, j \in \{1, \ldots, 2^s\}$ let

$$p_{i,j} = \Pr_{x \in G}[1 \xrightarrow{x}_Q i \text{ and } i \xrightarrow{x}_Q j]$$

and

$$q_{i,j} = \Pr_{y \in U_n}[1 \xrightarrow{y}_Q i \text{ and } i \xrightarrow{y}_Q j]$$

From (3.4.1) we have

$$\sum_{i \in \{1,\ldots,2^s\}} \sum_{j \in \{1,\ldots,2^s\}} |p_{i,j} - q_{i,j}| \geq \|Q(G^2) - Q(U_n^2)\|_1 > \varepsilon'$$

Therefore, there exist $i^*, j^* \in \{1, \ldots, 2^s\}$ such that

$$|p_{i^*,j^*} - q_{i^*,j^*}| > \varepsilon'/2^{2s}$$

Let Q' be a FSM with state space $\{1, \ldots, 2^s\} \times \{1, \ldots, 2^s\}$, and alphabet $\{0,1\}$. We define the transition function of Q' so that for any $b \in \{0,1\}$ and for any $i, j \in \{1, \ldots, 2^s\}$, we have $(i,j) \xrightarrow{b}_{Q'} (i', j')$ iff $i \xrightarrow{b}_Q i'$ and $j \xrightarrow{b}_Q j'$. We set the initial state of Q' to be $(1, i^*)$. We have

$$\begin{aligned}
\|Q'(G) - Q'(U_n)\|_1 &= \sum_{i,j \in \{1,\ldots,2^s\}} \left| \Pr_{x \in G}[(1,i^*) \xrightarrow{x}_{Q'} (i,j)] - \Pr_{y \in U_n}[(1,i^*) \xrightarrow{y}_{Q'} (i,j)] \right| \\
&\geq \left| \Pr_{x \in G}[(1,i^*) \xrightarrow{x}_{Q'} (i^*,j^*)] - \Pr_{y \in U_n}[(1,i^*) \xrightarrow{y}_{Q'} (i^*,j^*)] \right| \\
&= \left| \Pr_{x \in G}[1 \xrightarrow{x}_Q i^* \text{ and } i^* \xrightarrow{x}_Q j^*] - \Pr_{y \in U_n}[1 \xrightarrow{y}_Q i^* \text{ and } i^* \xrightarrow{y}_Q j^*] \right| \\
&= |p_{i^*,j^*} - q_{i^*,j^*}| \\
&> \varepsilon'/2^{2s} \\
&= \varepsilon
\end{aligned}$$

Therefore, G is not a PRG against FSMs of space $2s$ with parameter ε. □

Note that every time we apply Lemma 3.4.4 on a PRG G we get a new PRG G' which is secure against twice as many passes as G, with slightly worse space and error parameters. By repeating $O(\log \log n)$ times, we obtain Theorem 3.4.5 which we restate below.

Theorem 3.4.5. *BPL*[polylog] ⊆ *QuasiP.*

Breaking Nisan's PRG with polynomially many passes

Let n be the length of the output of Nisan's PRG. Fix $c > 0$ and the seed length to be $2^{c\sqrt{\log n}}$. In Theorem 3.4.3 we stated a corollary of the theorem from [Nis92], by choosing the parameters appropriately. We used Theorem 3.4.3 to obtain a positive result. However, for breaking Nisan's PRG (with polynomially many passes) we consider its general form. Note, that our adversary overwhelmingly breaks the security properties (stated for one-pass) of Nisan's PRG. In particular, for seed length $2^{c\sqrt{\log n}}$ instead of a $2^{c'\sqrt{\log n}}$ space, $c' > 0$, distinguisher we present a *logspace* one. Furthermore, we distinguish with probability much bigger than $2^{-c'\sqrt{\log n}}$.

Let us review the description of Nisan's PRG.

Let H be a universal family of hash functions $h : \{0,1\}^N \to \{0,1\}^N$, for some N. We are going to determine all parameters after the description of the PRG. For every integer $k \geq 0$ define the generator

$$G_k : \{0,1\}^N \times H^k \to (\{0,1\}^N)^{2^k}$$

where G_k is defined recursively as follows:

- $G_0(x) = x$, and
- $G_m(x, h_1, \ldots, h_m) = G_{m-1}(x, h_1, \ldots, h_{m-1}) \circ G_{m-1}(h_m(x), h_1, \ldots, h_{m-1})$, where $\circ$ denotes concatenation of strings.

Nisan shows that such a PRG is secure against FSMs that make a single pass over the input. He uses an explicit efficiently computable family H of affine functions $h : GF(2)^N \to GF(2)^N$. A function h is affine if there is a linear function f_h and a $b \in GF(2)^N$ such that $h(x) = f_h(x)+b$ for all $x \in GF(2)^N$. Nisan's [Nis92] family H has the property that each $h \in H$ can be described in $O(N)$ bits.

Additional terminology and notation. The output of the PRG consists of *blocks*, each of which is an N-bit binary string, and it corresponds to evaluating x on some composition of functions from H. Let $h_{i_1,\ldots,i_l} = h_{i_1} \circ \ldots \circ h_{i_l}$, where $\circ$ denotes function composition. By definition, in $G_k(x, h_1, \ldots, h_k)$ each block will be the composition of $h_{i_1} \circ h_{i_2} \circ \cdots \circ h_{i_l}$ evaluated on x, where $i_1 < i_2 < \cdots < l_l$. By definition, given the output y of the PRG on some x with family $H = \{h_1, \ldots, h_k\}$, an increasing sequence $\langle i_1, i_2, \ldots i_l \rangle$, $i_j \in \{1, \ldots, k\}$ uniquely determines the position of $h_{i_1,\ldots,i_l}(x)$ in y.

The parameters. The output length is $n = 2^k N$. Therefore, $k = O(\log n)$. For a family H as in [Nis92], the seed length is $O(kN)$. Therefore, for seed length $2^{c\sqrt{\log n}}$ we have that $N = O(2^{c\sqrt{\log n}})$.

The main idea. Suppose that the string y is the output of Nisan's PRG, with $H = \{h_1, \ldots, h_k\}$ the family of affine functions and some x. We will describe a test that every such y passes, but almost every string fails to pass. Note that to perform the test we do not need to know the actual h_i. The given $\langle i_1, \ldots, i_l \rangle$ is sufficient to determine the position of $h_{i_1,\ldots,i_l}(x)$ on y.

Here is a property that our test will use. We treat every block (which is in particular a binary substring of length N) of y as a vector in $GF(2)^N$. Let $y_1, \ldots, y_l$ be an even number of blocks in y and $h : GF(2)^N \to GF(2)^N$ an affine function. If $y_1, \ldots, y_l$ are linearly dependent[6] then $h(y_1) + \cdots + h(y_l) = 0$. Furthermore, suppose that $h := h_1 \in H$ and h_1 is not used in the composed functions determining the y_i's. Recall that for every block y_i we know the position of $h(y_i)$ on y.

Therefore, the output of Nisan's PRG y passes the following test: find an even number of linearly independent blocks $y_1, \ldots, y_l$ indexed by the h_i's where $i > 1$, and check whether the sum of the blocks corresponding to $h_1(y_i)$ evaluate to 0. Observe that to perform this test we don't have to know the actual h_i's. We refer to the h_i's just for determining the positions of the corresponding blocks.

Observe that if y is random then the probability that at least 2 vectors corresponding to N-bit substrings of y sum up to 0 is 2^{-N}.

The logspace distinguisher. Let y be the input to the distinguisher.

1. Let $I_1 := \{2, 3, \ldots, \frac{k}{2}\}$, and $I_2 := \{\frac{k}{2} + 1, \ldots, k\}$. Use I_1 and I_2 to index block positions. Let H_1 be the first $N + 1$ blocks indexed by $\langle i_1, \ldots, i_l \rangle$, $1 < i_1 < \ldots < i_l$, with $i_j \in I_1$. Similarly, we define H_2 where the indices are determined using I_2. Hence, in both cases for y (either y is the output of Nisan's PRG on random H and x, or y is a random string) we have that (i) $H_1 \cap H_2 = \emptyset$ with probability $\geq 1 - 2^{-\Omega(N)}$, and (ii) each of the sets is larger than N, which guarantees that the vectors in H_1 are linearly dependent, and the same holds for H_2.

2. Use [Mul87] (see below) to find in space $O(\log n)$ a set of linearly dependent vectors in H_1: $y_1^{(1)} + y_2^{(1)} + \ldots + y_{j_1}^{(1)} = 0$, and a set of linearly dependent vectors in H_2: $y_1^{(2)} + y_2^{(2)} + \ldots + y_{j_2}^{(2)} = 0$. As usual in space-bounded computation, by this we mean that each time we can (re)compute the indices of the vectors on the input tape.

3. If j_1 is even, accept iff $\sum_{j=1}^{j_1} \alpha_{y_j^{(1)}} = 0$, where $\alpha_{y_j^{(1)}}$ is the block indexed by the composition of h_1 with the composed functions corresponding to $y_j^{(1)}$. Similarly, if j_2 is even. Else, if both j_1 and j_2 are odd then $j_1 + j_2$ is even, and accept iff $\sum_{j=1}^{j_1} \alpha_{y_j^{(1)}} + \sum_{j=1}^{j_2} \alpha_{y_j^{(2)}} = 0$.

[6] For an even l, $h(y_1) + \cdots + h(y_l) = f_h(y_1) + \cdots + f_h(y_l) = f_h(y_1 + \cdots + y_l)$. In particular, $y_1 + \cdots + y_l = 0 \implies h(y_1) + \cdots + h(y_l) = 0$, since f_h is linear.

Clearly, if the input is random then the probability that the adversary accepts is $2^{-\Omega(N)}$. On the other hand, if the input is pseudorandom then with probability at least $1 - 2^{-\Omega(N)}$ we have that (i) $H_1 \cap H_2 = \emptyset$, and (ii) all vectors in $H_1 \cup H_2$ are non-zero. Conditioned on these two events, the adversary accepts with probability 1.

Finding the linear dependencies. Let NonSingular-Equations(n) be the problem of solving a non-singular $n \times n$ system of linear equations [BGH82]. From [Mul87] we have.

Theorem 3.4.6 (Mulmuley [Mul87])**.** NonSingular-Equations(n) *can be computed in the functional analog of the uniform* NC^2 *(i.e. the circuits can have multiple outputs).*

Consider the following procedure. Fix $y \in H_1$ and let A be the $N \times N$ matrix with columns $\{z \in H_1 - \{y\}\}$. If $y \neq 0$, then every solution of the system $Ax = y$ is non-zero. Since the set of vectors in H_1 is linearly dependent, there exists at least one y such that the system has a solution. This solution gives us the required set of vectors $y_1^{(1)} + \ldots + y_{j_1}^{(1)} = 0$. Similarly for H_2.

Recall that $\mathsf{NC}^2 \subseteq \mathsf{DSPACE}(\log^2 n)$. However, when solving the linear system $Ax = y$ in the above procedure, the input length is not n, but polynomial in $N = O(2^{c\sqrt{\log n}})$. Therefore, by Theorem 3.4.6, the above procedure can be implemented in space $O(\log n)$ and thus we obtain the following theorem. Note that the length of the advice is $O(2^{c\sqrt{\log n}})$.

Theorem 3.4.7. *Let $c > 0$. There exists $k > 0$, such that Nisan's PRG with seed of length $2^{c(\sqrt{\log n})}$ is not secure against logspace machines that make n^k passes over the input.*

3.4.2 The expected number of passes for probabilistic Verifier Stack Machines

We also consider the variant of probabilistic vSMs where the number of passes is an average-case resource. For RPdL the number of passes over the random tape is a worst-case resource, similar to the time-resource in the definition of RP and BPP. We define the hierarchy $\mathsf{RPdL}^{\mathbb{E}}$ same as RPdL where each level is defined with respect to the expected number of passes over the random tape. Contrast this to Randomness Compilers where a similar definition would be less natural. This shows an intuitive advantage of the vSM model.

Using a simple averaging argument we obtain that the first polynomially many levels of both hierarchies RPdL and $\mathsf{RPdL}^{\mathbb{E}}$ coincide. This is a corollary of the following lemma.

Lemma 3.4.8. $\mathit{RPdL}[r(n)] \subseteq \mathit{RPdL}^{\mathbb{E}}[r(n)] \subseteq \mathit{RPdL}[O(r(n))]$*, where* $r(n) = n^{O(1)}$*.*

The proof of this lemma follows by standard arguments. In particular, for the non-trivial inclusion, we simulate the machine where the expected number of passes is $r(n)$ say for $4r(n)$ steps, and then apply Markov's inequality.

3.5 Variants of the model: non-deterministic variants

We consider two nondeterministic variants of the model. The first is a vSM with a nondeterministic tape. In this case a certain compressibility property of NP-witnesses implies that two passes over the (nondeterministic) auxiliary tape suffice to do all of NP. The other variant is vTMs with a nondeterministic tape. In this case we show a tight connection with deciding width parametrizations of SAT, and in particular with SAT of fixed pathwidth.

3.5.1 NPdL[2] = NP

We show the following theorem. Not surprisingly, this intuitively implies that derandomizing the second level of the RPdL is non-trivial, in the sense that the boundedness of error is relevant.

Theorem 3.5.1. *The non-deterministic hierarchy collapses to level* 2*. More specifically,* $\mathsf{NPdL}[2] = \mathsf{NP}$*. Also,* $\mathsf{NPdL}[1] = \mathsf{P}$*.*

Proof. $\mathsf{NPdL}[1] = \mathsf{P}$ follows directly by Theorem 1 p.7 [Coo71].

We show that $\mathsf{NPdL}[2] = \mathsf{NP}$. By the NP-completeness of 3-SAT under many-to-one logspace reductions, it suffices to decide 3-SAT in $\mathsf{NPdL}[2]$, since $\mathsf{NPdL}[2]$ is closed under logspace reductions.

We describe a non-deterministic SM for 3-SAT. Let ϕ be the 3-CNF formula given in the input, with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Let τ be a truth assignment identified by the string $\langle \tau(x_1), \ldots, \tau(x_n) \rangle$ and let τ^R denote the reversed string. Simultaneously, we: (i) check the truth assignment is a satisfying one; and (ii) check that the external tape is of the form $\underbrace{\langle \tau, \tau^R, \ldots, \tau, \tau^R, \tau, \tau^R \rangle}_{m \text{ times}}$. Associate with the first clause C_1 the first copy of the truth assignment τ, with C_2 the second copy τ^R, and so on up to C_m.

To check (i) we do not use the stack. Verify that τ satisfies C_1, τ^R satisfies C_2 and so on. This verification can be done in one scan over the non-deterministic tape, just using the logspace worktape.

To check (ii) we use the stack and the worktape as follows. Suppose that the external memory contains the strings $w_1, \ldots w_{n-1}, w_n$. In the first scan the machine uses the stack to verify that $w_1 = w_2^R$, ..., $w_{n-3} = w_{n-2}^R$, and $w_{n-1} = w_n^R$. When the head reverses it uses the stack to verify that $w_{n-1} = w_{n-2}^R$ and so on. Intuitively, we first verify equality between mutually exclusive pairs and then we "link" them by verifying equality among the pairs. □

We remark that a similar argument shows that $\mathsf{PPdL}[2] = \mathsf{PP}$.

3.5.2 Verifying NP-witnesses with limited access to the witness

Our motivation for studying vTMs with non-deterministic tape goes beyond the study of yet-another-variant of the vSM model. We aim to understand the complexity of problems in NP, when the access to the NP-witness is parametrized. In a different direction, we aim to syntactically characterize the decision complexity of width-parametrizations of SAT. It turns out that these two are very well related.

Width-parametrizations of SAT are studied as a case of parametrized complexity (cf. [FG06]). There is an extensive body of work, too large to be cited here (e.g. see [Sze03] for an older survey). We characterize the complexity of SAT instances with path-decompositions of width $w(n)$. Although pathwidth is the most restrictive among the studied width-parametrizations of SAT, the most time-efficient algorithms known for such SAT instances run in time $2^{\Omega(w(n))}$, even when the path-decomposition is given in the input. We wish to better understand the decision complexity of SAT instances of width $\omega(\log n)$. We provide an exact correspondence between $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$, the problem of SAT instances with given path decomposition of width $w(n)$, and $\mathsf{NL}[r(n)]$, the class of problems decided by logspace Turing Machines with at most $r(n)$ passes over the polynomially long nondeterministic tape. In particular, we show that $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$ is hard for $\mathsf{NL}[\frac{w(n)}{\log n}]$ under logspace reductions. When $\mathsf{NL}[\frac{w(n)}{\log n}]$ is closed under logspace reductions, which is the case for the most interesting $w(n)$'s, we show that $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$ is also complete.

Note that by parametrizing on the number of passes over the non-deterministic tape we define a hierarchy between NL and NP. We denote by $\mathsf{NL}[r(n)]$ the class of sets accepted by vTMs where the head of the nondeterministic tape is allowed to reverse $r(n) - 1$ times, i.e. making $r(n)$ passes. Thus, $\mathsf{NL}[1] = \mathsf{NL}$ and $\mathsf{NL}[n^{O(1)}] := \cup_{c>0}\mathsf{NL}[n^c] = \mathsf{NP}$. Note that other than the nondeterministic tape, all other tapes have an unbounded number of reversals, as in a standard logspace TM.

Motivation for width-parametrizations of SAT. A common way to consider subproblems of SAT is to parametrize the input according to some graph-theoretic width-parameter measured on a graph associated with the input formula. The *incidence graph* is a canonical example of such a graph (see below for a definition). Treewidth, pathwidth, cliquewidth and so on, are width parameters for CNF instances, extensively studied both in theoretical and empirical research. This line of research gives algorithms (for inputs with restricted parameters) with running times significantly better than that of exhaustive search. Different parametrizations of SAT have been studied under different settings, but in general no algorithms faster than $2^{O(w(n))}$ are known, even when the width $w(n)$ corresponds to pathwidth, the most restrictive among the studied width-parameters. We observe that the Exponential Time Hypothesis (ETH), a

strong complexity assumption, can be used to obtain lower bounds on the time complexity of SAT instances of pathwidth $\omega(\log n)$. One motivating question is whether we can relate the non-existence of polytime algorithms for such SAT instances with complexity assumptions weaker than the ETH.

We denote by $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$ the set of satisfiable CNFs with given path decompositions of width $w(n)$, where n is the input length (definitions given later on in this section). By providing the path decomposition in the input in some sense we mod-out the difficulty of computing the pathwidth.

Conjecture. Let $w(n) = \omega(\log n)$. Then, $\mathsf{SAT}_{\mathsf{pw}}[w(n)] \notin \mathsf{P}$.

Conditioned on ETH this conjecture becomes true (see below). ETH [IPZ98] states that the N variable 3-SAT can't be solved in time $2^{o(N)}$. Making further progress seems difficult, partly because the precise complexity of $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$ is unknown. In [GP08] it is observed that for k-CNFs with path decompositions of width $w(n)$ the $2^{O(w(n))}$ running time can be improved to nondeterministic space $O(w(n))$. This improvement is far from an exact characterization, since $\mathsf{NSPACE}(w(n))$ is believed to contain sets not in NP, if $w(n) \in \omega(\log n)$. We use elementary techniques to precisely characterize the complexity of $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$ by identifying a correspondence between two seemingly unrelated concepts.

Regarding our motivating conjecture there are some very interesting works for similar questions, in particular for treewidth parametrized CSPs and for CLIQUE [FK97, Gro03, Mar07]. Also, understanding the power of vTMs is of independent interest. For example, for which $r(n) > 1$ is $\mathsf{NL}[r(n)] \subseteq \mathsf{P}$? We relate $\mathsf{NL}[\cdot]$ and $\mathsf{SAT}_{\mathsf{pw}}[\cdot]$ in a very strong sense.

Theorem 3.5.2. *Let $w(n)$ be constructible in space $O(\log n)$. Then, $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$ is complete for $\mathsf{NL}[\frac{w(n)}{\log n}]$.*

Although, the containment and hardness directions find applications on their own, the completeness makes sense for classes $\mathsf{NL}[r(n)]$ closed under logspace reductions. As we will see this is the case for the most interesting and useful $r(n)$'s.

Later on we list "Implications of the main Theorem 3.5.2". For example, based on the interplay between $\mathsf{SAT}_{\mathsf{pw}}$ and vTMs we obtain lower bounds on the number of nondeterministic tape head reversals for vTMs.

Remark 3.5.3. Several details matter for the exact characterization to work. For instance, it is crucial that the parameter is the number of passes over the nondeterministic tape and not some other measure of "use" of nondeterminism. Furthermore, the same characterization wouldn't work for treewidth, since $\mathsf{SAT}_{\mathsf{pw}}[\log n]$ is complete for NL, but we conjecture that the same problem for treewidth is complete for LOGCFL (one can show containment [GP08]), and it is thought that $\mathsf{NL} \subsetneq \mathsf{LOGCFL}$.

Definitions, notation, and preliminary results

Let ϕ be a CNF formula with variables $L := \{x_1, \ldots, x_N\}$ and clauses $R := \{C_1, \ldots, C_m\}$. The *incidence graph of* ϕ is a bipartite graph with bipartization (L, R), and there is an edge between a *variable-vertex* $x_i \in L$ and a *clause-vertex* $C_j \in R$, iff x_i appears (signed) in C_j. Path and tree decompositions are defined over arbitrary graphs as follows. A *tree decomposition of a graph* $G = (V, E)$ is the pair $\tau := (\{X_i : i \in I\}, T = (I, F))$, such that $X_i \subseteq V$, $\cup X_i = V$, T is a tree such that: (i) for every $\{u, v\} \in E$ there exists $i \in I$ such that $u, v \in X_i$, (ii) for every $u \in V$ let $X_{i_1}, \ldots, X_{i_k}$ be the collection of all sets containing u, then the subgraph of T induced by $i_1, \ldots, i_k \in I$ is connected (i.e. it is a tree). The *treewidth of* G is the minimum width over all decompositions, where the *width* of a decomposition is $\max_i |X_i| - 1$. By restricting the tree T to a path we define the *path decomposition*, and the *pathwidth* of G. It is well-known that for a graph G, $\mathsf{tw}(G) \leq \mathsf{pw}(G) \leq (\log |V|)\mathsf{tw}(G)$. Finally, we consider ordered k-CNFs Φ each being a sequence of clauses each consisting of at most k literals. For the given ordering the distance of a variable x is the index of the clause of the last appearance of x minus the first appearance of x. The *diameter of* Φ, $\mathsf{diam}(\Phi)$ is the maximum distance over its variables. The relevant computational problems are given below.

Conventions. All reductions are many-to-one and logspace. As usual we denote by n the length of the input. N is the number of variables in a CNF and M the number of clauses. All widths are functions of n. We only consider non-decreasing functions $w, r : \mathbb{Z}^+ \to \mathbb{Z}^+$ constructible in logarithmic space.

Problem: $\mathrm{SAT}_{\mathsf{pw}}[w(n)]$
Input: A CNF formula ϕ together with a path decomposition of the incidence graph of width $\leq w(n)$.
Output: Is ϕ satisfiable?

Problem: $\mathrm{SAT}_{\mathsf{diam}}[w(n)]$
Input: An ordered CNF formula Φ where $\mathsf{diam}(\Phi) \leq w(n)$.
Output: Is Φ satisfiable?

These problems are inter-reducible within logspace (see Lemma 3.5.4). Usually, we consider constant clause size CNFs for $\mathsf{SAT}_{\mathsf{diam}}$, and arbitrary CNFs for our central problem $\mathsf{SAT}_{\mathsf{pw}}$.

Lemma 3.5.4. *$\mathsf{SAT}_{\mathsf{diam}}[w(n)]$, $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$, 3-$\mathsf{SAT}_{\mathsf{pw}}[w(n)]$, and $\mathsf{SAT}_{\mathsf{pw}}[cw(n)]$, $c > 0$ are inter-reducible.*

In [GP08] it is shown that 3-$\mathsf{SAT}_{\mathsf{pw}}[w(n)] \in \mathsf{NSPACE}(w(n))$, and that 3-$\mathsf{SAT}_{\mathsf{pw}}[w(n)]$ and 3-$\mathsf{SAT}_{\mathsf{diam}}[w(n)]$ are logspace reducible to each other. Also, Lemma 3.5.14 states that $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$ is reducible to 3-$\mathsf{SAT}_{\mathsf{pw}}[w(n)]$. The proof of Lemma 3.5.14 is given after the "Implications of the main Theorem 3.5.2". The above Lemma 3.5.4 follows by [GP08], Lemma 3.5.14, and by padding; where by padding we mean that we extend a propositional formula by adding new clauses each of which contains a single, fresh variable.

Remark 3.5.5. Lemma 3.5.4 is somewhat interesting since there is a family of 3-CNFs where every variable appears only a constant number of times, the formulas are of pathwidth $O(\log n)$ and every ordering of the clauses has diameter $\Omega(\frac{n}{\log n})$ [GP08]. This combinatorial exponential gap is being smoothed out through logspace transformations.

We conclude this part with a useful consequence of the Exponential Time Hypothesis (ETH). Recall that ETH states that the N-variable 3-SAT cannot be solved in time $2^{o(N)}$.

Lemma 3.5.6. *Let $w(n) = \omega(\log n)$. If ETH holds then 3-$\mathsf{SAT}_{\mathsf{pw}}[w(n)] \notin \mathsf{P}$.*

Proof. Suppose that there is a polytime algorithm for $\mathsf{SAT}_{\mathsf{pw}}[w(n)]$. Since for 3-CNFs, $M = \Theta(n \log n)$ there is an (also logspace constructible) superlogarithmic function on the number of clauses, $w'(M) = \omega(\log M)$, where there is polytime algorithm A that solves instances of pathwidth $w'(M)$. Given a 3-CNF ϕ we decide its satisfiability as follows. Pad the formula with single-fresh-variable clauses such that for the number of clauses M' of the new formula, we have $w'(M') \geq M$. Thus, we construct ϕ' using $2^{o(M)}$ new clauses, and the obvious path decomposition of ϕ' is of width $\leq w'(M')$. Hence, A decides satisfiability in time $2^{o(M)}$, which by the Sparsification Lemma[7] [IPZ98] implies an $2^{o(N)}$ algorithm, contradicting ETH. □

Here is a simple but very interesting implication of the above.

Theorem 3.5.8. *$\mathsf{PolyLogSpace} \cap \mathsf{NP} \not\subseteq \mathsf{P}$, unless the ETH fails. Furthermore, there exists $\epsilon > 0$ such that 3-$\mathsf{SAT}_{\mathsf{pw}}[\log^{\epsilon} n] \in \mathsf{PolyLogSpace} \cap \mathsf{NP}$ and 3-$\mathsf{SAT}_{\mathsf{pw}}[\log^{\epsilon} n] \notin \mathsf{P}$, unless the ETH fails.*

Proof. From [GP08] we have that 3-$\mathsf{SAT}_{\mathsf{pw}}[\log^2 n] \in \mathsf{NSPACE}(\log^2 n) \subseteq \mathsf{DSPACE}(\log^4 n)$. By Lemma 3.5.6 we have that 3-$\mathsf{SAT}_{\mathsf{pw}}[\log^2 n] \notin \mathsf{P}$, unless ETH fails. □

[7] The precise statement of the Sparsification Lemma requires background on notation and definitions not necessary in our exposition. We will make use of the following corollary of the Sparsification Lemma [IPZ98].

Corollary 3.5.7 ([IPZ98])**.** *Let N, M denote the number of variables and clauses of k-CNF formulas, for an arbitrary fixed constant $k \in \mathbb{Z}$. If there exists an algorithm A that solves k-SAT in time $2^{o(M)}$ then there exists an algorithm A' that solves k-SAT in time $2^{o(N)}$.*

Implications of the main Theorem 3.5.2

The main result for non-deterministic vTMs is Theorem 3.5.2. There are a number of consequences of the containment (Lemma 3.5.15) and hardness (Lemma 3.5.16) directions of this theorem.

Let us begin with Corollary 3.5.9, which follows by the containment and hardness of Theorem 3.5.2, Lemma 3.5.4, and the $\mathsf{NSPACE}(w(n))$ algorithm for 3-$\mathsf{SAT}_{\mathsf{diam}}[w(n)]$ given in [GP08].

Corollary 3.5.9. *(i)* $\mathit{NL}[r(n)] = \mathit{NL}[cr(n)]$, $c > 0$, *and (ii)* $\mathit{NL}[r(n)] \subseteq \mathit{NSPACE}(r(n)\log n)$.

Proof.
(i) The non-trivial inclusion $\mathsf{NL}[cr(n)] \subseteq \mathsf{NL}[r(n)]$ is a corollary of the reduction (algorithm) in the proof of Lemma 3.5.16.

(ii) Let $L \in \mathsf{NL}[r(n)]$. By Theorem 3.5.2 L reduces to 3-$\mathsf{SAT}_{\mathsf{diam}}[r(n)\log n]$. By the algorithm in [GP08] we conclude that $L \in \mathsf{NSPACE}(r(n)\log n)$. □

Corollary 3.5.9 immediately implies the following.

Corollary 3.5.10. $\mathit{NL}[r(n)] \subseteq \mathit{NP} \cap \mathit{NSPACE}(r(n)\log n)$. *In particular,*
$\mathit{NL}[\mathsf{polylog}] := \cup_{k>0}\mathit{NL}[\log^k n] \subseteq \mathit{NP} \cap \mathit{PolyLogSpace}$.

Here are some further implications of Corollary 3.5.9.

As mentioned, the most interesting cases for $w(n)$ are those where $\mathsf{NL}[w(n)/\log n]$ is closed under logspace reductions; e.g. $\mathsf{NL}[1], \mathsf{NL}[\log\log n], \mathsf{NL}[\log^k n]$. The closure under logspace reductions is immediate by Corollary 3.5.9, which also implies that a constant number of passes over the nondeterministic tape can't get us outside NL.

Corollary 3.5.11. $\mathit{NL}[O(1)] := \cup_{i=1}^{\infty}\mathit{NL}[i] = \mathit{NL}$

Since $\mathsf{NL}[O(1)] = \mathsf{NL}$ and $\mathsf{NL}[n^{O(1)}] = \mathsf{NP}$, it is natural to ask what is the smallest $r(n)$ such that $\mathsf{NL}[r(n)] \supseteq \mathsf{P}$. Under a widely believed complexity assumption, Corollary 3.5.9 implies that it is not possible to capture P before NP.

Corollary 3.5.12. *There does not exist* $k > 0$ *such that* $\mathit{NL}[n^k] \supseteq \mathit{P}$, *unless* $\mathit{P} \subseteq \mathit{DSPACE}(n^{k'})$, *for some* $k' \in \mathbb{Z}^+$.

It follows by Corollary 3.5.9 and Theorem 3.5.2 that there are $r(n)$'s where $\mathsf{NL}[r(n)]$ is not closed under logspace reductions, modulo complexity assumptions. For example, if $\mathsf{NL}[n^\epsilon]$ is closed under logspace reductions then $\mathsf{NP} \subseteq \mathsf{NL}[n^\epsilon] \subseteq \mathsf{DSPACE}(n^{\epsilon'})$.

A perhaps surprising consequence of the relation between $\mathsf{SAT}_{\mathsf{pw}}[\cdot]$ and $\mathsf{NL}[\cdot]$ is Corollary 3.5.13, which follows by the containment direction of Theorem 3.5.2 and Lemma 3.5.6.

Corollary 3.5.13. *Let $r(n) = \omega(1)$. If ETH holds then $\mathsf{NL}[r(n)] \not\subseteq \mathsf{P}$.*

An example of an L in Corollary 3.5.13 is 3-$\mathsf{SAT}_{\mathsf{pw}}[\omega(\log n)]$ (note that Lemma 3.5.15 also holds for 3-$\mathsf{SAT}_{\mathsf{pw}}[w(n)]$).

Observe that we can get Theorem 3.5.8 directly from Corollaries 3.5.13 and 3.5.10. However, we believe that Corollary 3.5.13 is strictly stronger than Theorem 3.5.8, since we conjecture that $\mathsf{NL}[\mathsf{polylog}] \subsetneq \mathsf{NP} \cap \mathsf{PolyLogSpace}$.

Proof of Main Theorem and Lemma 3.5.14

Lemma 3.5.4 states that 3-$\mathsf{SAT}_{\mathsf{diam}}$ and $\mathsf{SAT}_{\mathsf{pw}}$ are inter-reducible, and thus Lemmas 3.5.15 and 3.5.16 imply Theorem 3.5.2. Lemma 3.5.14 allows us to restrict to 3-CNFs, and thus to conclude Lemma 3.5.4.

Lemma 3.5.14. *$SAT_{\mathsf{pw}}[w(n)]$ is reducible to 3-$SAT_{\mathsf{pw}}[3w(n)]$.*

Proof. Fix 3-CNF Φ of size n, and fix a path decomposition $P = \langle X_1, X_2, \ldots, X_\alpha \rangle$ of width $\leq w(n)$. Let $C = \{l_1, \ldots, l_k\}$ be a clause of Φ, and assume that $l_1, \ldots, l_k$ respects the ordering of the variables in P. We introduce new variables $y_1, \ldots, y_{k-1}$, and note that a truth assignment to the old variables satisfies C iff some extension to the new variables satisfies all $\{l_1, \neg y_1\}, \{y_1, l_2, \neg y_2\}, \ldots, \{y_{k-2}, l_{k-1}, \neg y_{k-1}\}, \{y_{k-1}, l_k\}$.

Consider the smallest s such that C appears in X_s together with some variable which appears in C. Remove C from every $X_{s'}$, where $s' < s$.

Let $m \geq s$ and let X_m contain clause C, and say that the literals $l_{i+1}, \ldots, l_{i+k_m}$ are the literals of C whose variables are in X_m and they are not in any earlier $X_{m'}$ that contains C.

Case $k_m = 0$: in this case all variables for C that are also in X_m have already appeared in some $X_{m'}$, $m' < m$ (in fact, since it's a path decomposition they appear in X_{m-1}). Then, we just add to X_m the last $y_{k'}$ corresponding to C which appears in X_{m-1}.

Case $k_m > 0$: replace X_m by the subpath $X_m^1, \ldots, X_m^{k_m}$ where X_m^j is obtained from X_m by removing C, adding the new variables from the new clause containing l_{i+j}, and adding that new clause as well.

It is not hard to verify that the following properties of the above transformation.

- Because of the rule that we only do the construction of the second case when we have newly appearing literals, this means that the final path decomposition will have length a polynomial in α and $w(n)$. Let us give an outline of an argument showing that this is true. Consider an X_m in the original P. Think of X_m as being the root of a leaf-ordered tree which represents the splittings of the initial X_m. The (ordered) leaves of the tree

correspond to what X_m will be replaced with; i.e. this tree is induced by the "splittings" that happen because of the "case $k_m > 0$". Then, since we are not splitting unless we have new literals we observe that this tree grows in depth along its leftmost path and at each level we have $O(w(n))$ nodes.

- It is easy to see that given the index of an X_j in the output and a clause C or a variable x, we can determine in logspace whether $C \in X_j$ or $x \in X_j$.
- This transformation results in a valid path decomposition of width $\leq 3w(n)$.

Note that the resulting formula is satisfiable iff Φ is satisfiable, and the theorem follows by padding. □

Lemma 3.5.15. *3-$SAT_{\mathsf{diam}}[w(n)] \in NL[\frac{w(n)}{\log n}]$, for $w(n) = \Omega(\log n)$.*

Proof. Let Φ be an ordered 3-CNF formula of size n and diameter $\leq w(n)$. We now present a nondeterministic logspace algorithm that in each pass over the nondeterministic tape verifies that $\frac{n}{w(n)} \log n$ new clauses are satisfied. Hence, Φ is covered in $\leq \frac{w(n)}{\log n}$ passes.

Consider an ordering of the variables of (the clause-ordered) Φ consistent with the ordering of the clauses. We view the nondeterministic tape as containing a truth assignment, where the i-th bit of the tape is the truth value of the i-th variable. By definition of diameter, if the j-th clause in the input has a variable which corresponds to the i-th bit on the nondeterministic tape, then the $(j + w(n))$-th clause contains only variables with index $\geq i$. For simplicity of presentation suppose that all divisions are for divisible integers. Partition the ordered clauses into consecutive blocks each containing $\log n$ clauses. The algorithm is iterative, and in each iteration it deals with exactly one such block which we call "effective block". Reserve space for a $\log n$ bit vector on the worktape. Each bit of this vector is associated with a clause in the "effective block". That is, the input tape is partitioned in blocks $b_1, b_2, \ldots, b_{n/\log n}$. We define the distance of two blocks to be the number of clauses between them. Two blocks of distance at least $w(n)$ do not share any variables. Partition the input tape further into super-blocks, each of size $w(n)$. That is, we have $\frac{n}{w(n)}$ super-blocks each containing $\alpha := \frac{w(n)}{\log n}$ blocks. That is, the blocks can be re-indexed as follows: $b_{1,1}, b_{1,2}, \ldots, b_{1,\alpha}, b_{2,1}, \ldots, b_{2,\alpha}, \ldots, b_{n/w(n),\alpha}$. Iteratively make passes over the nondeterministic tape, where in each pass $\frac{n}{w(n)}$ blocks are verified to be satisfiable, as follows.

In the odd numbered passes the nondeterministic head scans from the left to the right end of the nondeterministic tape, and symmetrically for the even numbered ones. We present the algorithm for an odd numbered iteration; symmetrically the algorithm works for the even numbered one. In the i-th iteration we verify the satisfiability of $b_{j,i}$'s, $j = 1, 2, \ldots, \frac{n}{w(n)}$. Start moving the head on the nondeterministic tape from left to right until we encounter the first

variable x that appears in $b_{1,i}$. Check which (if any) clauses of $b_{1,i}$ are satisfied by x and record this on the corresponding bits on the reserved $\log n$ bit vector on the worktape. Continue this way until the last variable in the block is read. If the reserved space shows that there is a clause not satisfied then reject. Else, the nondeterministic head has not read yet any variable from $b_{2,i}$. Move the nondeterministic head further and repeat for $b_{2,i}$ as for $b_{1,i}$; similarly, for $b_{3,i}, \ldots, b_{n/w(n),i}$.

This algorithm is well-defined since the truth values for the verification they are located at most $w(n)$ to the left and $w(n)$ to the right of each block. The correctness of this logspace algorithm is transparent in its description. □

An easy modification of the above algorithm makes the lemma true for 3-$\mathsf{SAT}_{\mathsf{pw}}[w(n)]$.

Lemma 3.5.16. *3-$SAT_{\mathsf{diam}}[w(n)]$ is hard for $\mathsf{NL}[\frac{w(n)}{\log n}]$, $w(n) = \Omega(\log n)$.*

Proof. Let $r(n) = \frac{w(n)}{\log n}$. Since $w(n)$ is computable in space $O(\log n)$, $r(n)$ is also logspace computable. Fix arbitrary $L \in \mathsf{NL}[r(n)]$ and let M be a 1-tape logspace vTM accepting L, which makes $r(n)$ passes over its nondeterministic tape, and is standardized as follows.
(i) M has a unique accepting configuration, (ii) the input head movement of M is oblivious (meaning that the position of the head is a function of the time step for the given input length), (iii) M is oblivious in the nondeterministic head movement and in fact it moves the nondeterministic head in each step in a left to right end (and right to left end) fashion. We can easily achieve (i), (ii) and make the nondeterministic head move oblivious by slowing down the computation. In particular, it is easy to make the nondeterministic head stay-put for a polynomial number of steps $p(n)$ in between moves, and make it move from the left to the right end of the polynomially $q(n)$-long tape and back. To standardize the machine to satisfy (iii) we proceed as follows. Construct a nondeterministic logspace TM verifying in a single pass whether its nondeterministic tape is of the form $\underbrace{b_1 b_1 \ldots b_1}_{p(n)\text{times}} \underbrace{b_2 b_2 \ldots b_2}_{p(n)\text{times}} \ldots \underbrace{b_{q(n)} b_{q(n)} \ldots b_{q(n)}}_{p(n)\text{times}}$ and reject if it is not. Composing this logspace machine with the previously standardized logspace machine we obtain a vTM M for L, satisfying properties (i), (ii), and (iii). On an input of length n, our standardize TM makes polynomially many $t(n)$ steps.

The hardness follows by modifying the construction in Cook's theorem for the NP-completeness of SAT (cf. [DK00] pp. 48-50). For simplicity, we initially deal only with the case where the number of passes $r(n) = 1$.

Case $r(n) = 1$: Let x be an input of length n. We wish to construct an ordered 3-CNF formula Φ of diameter $O(\log n)$ (which by padding is sufficient), such that Φ is satisfiable iff the M accepts x. We define a configuration γ to be a string which is the concatenation of the strings: (i) the $O(\log n)$ symbols of the worktape, encoding the worktape symbol which the head is on

by a special symbol, (ii) the state of M, and (iii) the value of the nondeterministic bit at the current position in the nondeterministic tape.

For our (standardized) vTM we wish to check whether there exists a sequence of configurations $\gamma_1 \vdash \gamma_2 \vdash \ldots \vdash \gamma_{t(n)}$, where γ_1 is the initial configuration on x, and $\gamma_{t(n)}$ is the accepting configuration. Our formula will have variables $y_{i,j,\alpha}$; we will enforce the constraints that $y_{i,j,\alpha} = 1$ iff symbol at position j of the configuration at step i is α. We first express in the usual way that γ_1 is the initial configuration. The main technical part is to enforce for $i = 1, \ldots, t(n)-1$ that $\gamma_i \vdash \gamma_{i+1}$. We can do this in the usual way, noting that the j-th symbol of γ_{i+1} only depends on a constant number of symbols of γ_i, since (by obliviousness) the position read by the input head is determined by i. It is also important to realize that we do not need to encode any constraint on the bit read on the nondeterministic tape since a new bit is read at each step, and since $r(n) = 1$ no bit is ever reread. Lastly, we express in the usual way that $\gamma_{t(n)}$ is the accepting configuration. To see that our ordered formula is of diameter $O(\log n)$, the main point is that the subformula needed to express $\gamma_i \vdash \gamma_{i+1}$ has $O(\log n)$ 3-literal clauses and uses only those variables associated with γ_i and γ_{i+1}; diameter $O(\log n)$ comes from the fact that we encode both $\gamma_i \vdash \gamma_{i+1}$ and $\gamma_{i+1} \vdash \gamma_{i+2}$.

Case arbitrary $r(n)$: If we perform the above reduction exactly as stated for $r(n) > 1$ we do not have a valid reduction, since we must enforce consistency between the different time steps in which the same nondeterministic bit is visited. If we modify the above construction such that we reserve the same variable name for the same nondeterministic bit, then this results in a formula of diameter $n^{\Omega(1)}$. Instead, we will always use the same variable name for the k-th nondeterministic bit, and we will write together all the subformulas describing those $r(n)$ transitions where the nondeterministic head moves between the k-th and $(k+1)$-th position (in either direction). We do this for $k = 1, \ldots, z(n)$ each time creating a subformula of diameter $O(r(n) \log n)$, where $z(n)$ is the polynomial length of the nondeterministic tape. This can also be expressed in a diagram, where we reorder the configurations denoting by $\gamma_{\mathsf{pass},\mathsf{symbol}}$ the configuration when the nondeterministic head does the number pass scan over the nondeterministic tape and it currently is at position symbol. Then, the computation starts by $\gamma_{1,1} \vdash \gamma_{1,2} \vdash \ldots \vdash \gamma_{1,z(n)}$ and the next configuration is $\gamma_{2,z(n)}$ since in the second pass the nondeterministic head moves from the right to the left end.

$$\begin{array}{ccccccc}
\gamma_{1,1} & \vdash & \gamma_{1,2} & \vdash & \dots & \vdash & \gamma_{1,z(n)} \\
\gamma_{2,1} & \dashv & \gamma_{2,2} & \dashv & \dots & \dashv & \gamma_{2,z(n)} \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \\
\gamma_{r(n),1} & \vdash & \gamma_{r(n),2} & \vdash & \dots & \vdash & \gamma_{r(n),z(n)}
\end{array}$$

Matrix describing the computation

$$\begin{pmatrix} \gamma_{1,k} \\ \gamma_{2,k+1} \\ \vdots \\ \gamma_{r(n),k} \end{pmatrix} \vdash \begin{pmatrix} \gamma_{1,k+1} \\ \gamma_{2,k} \\ \vdots \\ \gamma_{r(n),k+1} \end{pmatrix}$$

Configurations regarding the nondeterministic bits k and $k+1$

□

Remark 3.5.17. Standardizing the machine in the proof of Lemma 3.5.16 significantly simplifies the argument. An alternative way to proceed without this standardization involves renaming variables (e.g. by formulas of the form $x \leftrightarrow y$) and interpolating them in appropriate places (which is a non-obvious task).

Chapter 4

An approach to lower bounds

We show a simultaneous $n^{\Omega(1)}$ lower bound in the number of passes over the input tape and the space of a Stack Machine computing a problem in NC. This is the first lower bound of this type. Its novelty is technical and conceptual. Conceptually, it falls in our framework of proving lower bounds - aiming to circuit lower bounds for languages[1] in NC in a way orthogonal to previous approaches. Technically, this is the first lower bound for a streaming model which has an additional unbounded, read/write memory, restricted to be a stack. By *unbounded* we mean that we do not charge for accessing the stack. The argument is a Communication Complexity reduction from inner product, in the two-player model, to the computation of a Stack Machine that solves a variant of inner product (a special case of P-Eval). As a side result we obtain a polynomial separation between a streaming model with a stack where the passes on the external tape are unidirectional, and a streaming model with a stack where the head can reverse direction - i.e. similar to the very recent work of Magniez, Mathieu and Nayak [MMN10].

In this chapter we include only the technical developments for the lower bound. For motivation and discussion see Section 1.3. In Section 4.1 we give necessary definitions and notation. In Section 4.2 we discuss the technical difficulty and the main issues in proving the lower bound for Stack Machines. We also provide a comparison and show machine-model separation between this and previous work. The remaining three sections regard the proof of Theorem 4.3.1. In Section 4.3 we give an outline of the proof, and in Section 4.4 we list the main lemmas, and we outline their proofs through examples, remarks, and intuitive explanations. We conclude with the actual proof in Section 4.5.

4.1 Definitions and preliminaries

[1] Recall that logspace bounded Stack Machines with a single pass over the input can even compute problems in $\mathsf{P} - \mathsf{NC}$ (unless $\mathsf{exp} = \mathsf{PSPACE}$).

Communication Complexity. We consider the standard two-player Communication Complexity model, where each of the two players receives n bits, and they communicate to compute a boolean function $f : \{0,1\}^{2n} \to \{0,1\}$. The resource we measure is the number of communicated bits. Communication is being done through a shared blackboard (this generalizes to the p-player Number In Hand model). Deterministic protocols are always correct. For randomized protocols the model is extended with shared randomness and we require correctness with probability at least $2/3$. For a function f we denote by $D(f)$ the cost of the best deterministic protocol, and by $R(f)$ the cost of the best randomized protocol over all inputs and random strings. For a detailed description of the model see e.g. [KN97].

In chapters 4 and 5 we consider the following functions for which we are interested in their communication complexity. For a family of functions $\mathcal{F}$, we denote by $\mathcal{F}_{2,\text{ player's input size}} = \mathcal{F}_{2,n}$ the corresponding function for inputs of size n given to each of the 2 players. The constant 2 denotes the number of players - our model is a special case of the p-player Number In Hand model. We insist in denoting the number of players as a reminder that the input length is $2n$. When no confusion arises we write $\mathcal{F}$ instead of $\mathcal{F}_{2,n}$, and vice-versa.

Here is some initial notation. For a function $f : \{0,1\}^{2n} \to \{0,1\}$ we use the notation $f(x_1, x_2)$, where $|x_i| = n$ to denote that Player-i gets input x_i. For a string $y \in \{0,1\}^k$ we denote by $\mathsf{supp}(y) \subseteq \{1, \ldots, k\}$ the set of non-zero indices of y. When necessary we identify subsets of $\{1, \ldots, k\}$ by the support of strings in $\{0,1\}^k$.

- *Inner product*: $\text{IP}^{\mathbb{F}}_{2,n} : \mathbb{F}^{2n} \to \mathbb{F}$, is the standard inner product over the field $\mathbb{F}$.
- *String equality*: $\textsc{Equal}_{2,n} : \{0,1\}^{2n} \to \{0,1\}$, where $\textsc{Equal}_{2,n}(x_1, x_2) = 1$ iff $x_1 = x_2$.
- *Set intersection*: $\textsc{SetInt}_{2,n} : \{0,1\}^{2n} \to \{0,1\}$, where $\textsc{SetInt}_{2,n}(x_1, x_2) = 1$ iff $\mathsf{supp}(x_1) \cap \mathsf{supp}(x_2) \neq \emptyset$.

Here are the Communication Complexity lower bounds we use in this and the next chapter. For a restricted, promise version of $\textsc{SetInt}_{2,n}$ [CKS03] shows in the case of 2 players a linear lower bound for the randomized complexity. Therefore, $R(\textsc{SetInt}_{2,n}) = \Omega(n)$. A simple reduction from this promise version to inner product over arbitrary $\mathbb{F}$, shows that $R(\text{IP}^{\mathbb{F}}_{2,n}) = \Omega(n)$. Also, it is well-known (e.g. [KN97]) that $D(\textsc{Equal}_{2,n}) \geq n-1$. For clarity of presentation we restrict to the case where $\mathbb{F} = \mathsf{GF}(2)$. However, our results hold for every $\mathbb{F}$ with an efficient encoding.

Lift-permute-combine. Our lower bound is for a variant of inner product, which is again an "inner product problem" where the input is presented in a special way. To that end, during the reduction the players construct a lifting of the original inner product instance, by constructing

m instances of IP which they permute and combine appropriately. We introduce notation for the concept of lift-permute-combine in more generality.

Definition 4.1.1. Consider a *base function* $f : \{0,1\}^{2N} \to \{0,1\}$. In the communication complexity model Player-i gets x_i, from an input $(x_1, x_2) \in \{0,1\}^{2N}$. We denote by $\mathsf{Sym}(m)$ the symmetric group on $\{1, 2, \ldots, m\}$. Let $\pi \in \mathsf{Sym}(m)$, for some parameters N and $m := m(N)$. The $(2mN)$-bit function $f^{\oplus,\pi} : \{0,1\}^{2mN} \to \{0,1\}$ is defined on input $x = (x_1, x_2) \in \{0,1\}^{2mN}$, where $x_i = (x_{i,1}, \ldots, x_{i,m}) \in \{0,1\}^{Nm}$, where each $|x_{i,j}| = N$, $i = 1, 2$, $j = 1, \ldots, m$, as follows: $f^{\oplus,\pi}(x) = \bigoplus_{j=1}^{m} f(x_{[j],\pi})$, where $x_{[j],\pi} = (x_{1,j}, x_{2,\pi(j)})$. When no confusion arises instead of $x_{[j],\pi}$ we write $x_{[j]}$.

Note that in a more general form of this definition, instead of $\oplus$ one could have used some other *combiner* function. Observe that when $f = \mathrm{IP}_{2,N}$ then the lifting of an inner product instance is by definition an inner product instance, where the input is presented in a permuted way.

Notational remark. Observe that given N, π, $n = mN$ the notation $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$ determines each of the following parameters: N, m, n, since $\pi \in \mathsf{Sym}(m)$.

Sortedness. Due to the bidirectional head-move our argument relies on permutations π of small sortedness. The *sortedness of a permutation* π *on* $\{1, \ldots, m\}$ is the maximum length over all increasing and decreasing subsequences of $\langle \pi(1), \ldots, \pi(m) \rangle$. By e.g. [ES35] it is known that $\sqrt{m} < \mathsf{sortedness}(\pi)$, and that there exists π such that $\mathsf{sortedness}(\pi) = O(\sqrt{m})$.

Here are some conventions and additional notation. From this point we assume that for every m the permutation π is such that $\mathsf{sortedness}(\pi) = O(\sqrt{m})$. Let $d = O(r(2mN)\log(2r(2mN))/\sqrt{m})$. In what follows the choice of r and m will be such that $r(2mN)\log(2r(2mN)) = o(\sqrt{m})$.

Machine models. We consider Stack Machines, parametrized on the number of passes over their input, as opposed to Chapter 3 where we parametrized over the auxiliary tape. Our main lower bound holds for the more general model of randomized, non-uniform Stack Machines. A *randomized Stack Machine* is a Stack Machine that flips coins (not to be confused with the vSMs with an auxiliary random tape of the previous section). A non-uniform, $s(n)$ space-bounded Stack Machine can be identified by a family of two-way finite state machines each with $2^{O(s(n))}$ states for input of length n. We write (r, s)-*SM* to denote an $s(n)$ space-bounded Stack Machine that makes at most $r(n)$ passes over its input. We also write (r, s, t)-*streaming algorithm* to denote a Turing Machine with workspace $s(n)$ (on which we do not count head-reversals), it has at most t external tapes, and it makes in total at most $r(n)$ passes over its external tapes.

4.2 Motivation and comparison to previous work

4.2.1 Warm-up: a streaming lower bound for IP

The following example demonstrates a technically straightforward application of Communication Complexity to streaming lower bounds. This is the only section of this chapter where we consider space-bounded Turing Machines without a stack. A corollary of Lemma 4.2.1 is that a logspace bounded Turing Machine requires $n^{\Omega(1)}$ passes over the input to compute $\mathrm{IP}_{2,n}$. Technically, we proceed by reducing IP to the computation of a space-bounded TM.

Lemma 4.2.1. *Let M be an $s(n)$-space bounded Turing Machine correct for $\mathrm{IP}_{2,n}$. Then, M requires $\Omega(\frac{n}{s(n)})$ passes over the input to solve $\mathrm{IP}_{2,n}$. Similarly, for $\textsc{SetInt}_{2,n}$.*

Proof. Consider $\mathrm{IP}_{2,n}$. This is a standard argument where we construct a two-player Communication Complexity protocol using the computation of M. Let $n \in \mathbb{Z}^+$. Player-1 gets input $x \in \{0,1\}^n$ and Player-2 $y \in \{0,1\}^n$, for the inner product $\langle x, y\rangle$. The protocol proceeds in rounds, where in each round the two players alternate as follows. Player-1 simulates M up until the point where the head crosses to the y part of the input. Note that the configuration of the machine is $O(s(n))$ bits long. Player-1 sends the $O(s(n))$ bits to Player-2 who continues the simulation until the input head crosses back to the x part of the input. Then, Player-2 sends the $O(s(n))$ bits to Player-1, and the protocol continues with the next round. The head crosses the middle of the tape at most $r(n)$ times. By the lower bound of $\mathrm{IP}_{2,n}$ we know that every protocol requires $\Omega(n)$ bits and thus $r(n) = \Omega(n/s(n))$.

Similarly for $\textsc{SetInt}_{2,n}$. □

What changes in presence of an unbounded stack? When we add a stack, unbounded in the sense that we do not charge passes over the stack, the above argument fails dramatically.

1. $\mathrm{IP}_{2,n}$ *can be solved with two passes (i.e. one head reversal).* Consider a Stack Machine M that on input x, y works as follows. In the first pass it pushes x to the stack. That is, at the end of the first pass the stack contains x^R (the string x reversed). In the second pass the head moves from the right end to the left; i.e. during this move it reads y^R. Hence, on the second pass M pops the stack symbols one by one and it computes $\langle x^R, y^R\rangle = \langle x, r\rangle$.

2. *The configuration of the machine is too big.* It is not only the case that $\mathrm{IP}_{2,n}$ cannot be an example for a lower bound based on the proof template of Lemma 4.2.1. Rather, for every problem Π the above technique fails. The reason is that the configuration of the SM is much bigger than $O(s(n))$, since the stack context can be polynomially large. In other words, no lower bound is possible using this proof technique.

4.2.2 Stack Machines compared to other streaming models

Our main technical contribution deals with the issue mentioned in the previous section. In the literature the basic streaming model [AMS96] has been extended in several ways, e.g. [GS05, BHN08]. It is reasonable to ask whether lower bounds for Stack Machines can be derived from streaming lower bounds on machines with multiple external tapes. We can think of two ways to "derive" lower bounds for Stack Machines from previous streaming lower bounds. The first is by simulating a Stack Machine with the streaming model with multiple tapes. The second is conceptual, and it refers to modifying existing proofs. We answer the first interpretation of the question in the negative. As for the second, we give strong technical evidence that no such modification is possible in an obvious way.

Comparison of computational models. Is it possible to perform useful computation using the stack without moving the input head? Intuitively, the proof of the following lemma is related to a positive answer to this question.

Lemma 4.2.2 (The power of one-way SMs)**.** *Let $m \in \mathbb{Z}^+$. There exists $L \in \mathsf{P}$ such that (i) there is a SM deciding L with one pass over the input, and (ii) every TM M with a logspace work tape (on which we don't count passes) and a constant number of tapes on which we count passes, M cannot decide L with $\log^m n$ passes, unless $\mathsf{E} \subseteq \mathsf{PSPACE}$.*

Proof. Consider the canonical complete language for E under log-lin reductions, $U := \{M\#w\#^{k|w|} \mid M(w) \text{ accepts within time } 2^{(k+1)|w|}\}$. Let the tally set[2] $T_U := \{0^{|x|} \mid x \in U\}$.

Fact 4.2.3.

1. *If there exists $L \in (\mathsf{E} - \mathsf{PSPACE})$, then U requires super-polynomial space.*
2. *$T_U \in \mathsf{P}$*
3. *If there exists $L \in (\mathsf{E} - \mathsf{PSPACE})$, then T_U requires super-polylogarithmic space.*

Proof. Suppose $U \in \mathsf{PSPACE}$ and arbitrary $L \in \mathsf{DTIME}(2^{k'n})$, witnessed by a TM M, i.e. $\mathcal{L}(M) = L$. An obvious simulation (by considering the projection of U when the machine is M and $k = k'$) shows that L also requires polynomial space. T_U is trivially in P since it is the unary representation of the strings in $U \in \mathsf{E}$. The last bullet follows by the first, and the fact that T_U is the unary representation of U. □

[2] A language L is tally if $L \subseteq \{0\}^*$.

Since T_U is a polytime tally set, it can be decided by an one-way SM (Proposition 2.1 [All89]). Suppose that there is $L \in (\mathsf{E} - \mathsf{PSPACE})$ and that T_U can be decided by a TM, with a logspace working tape (on which we don't count passes), and a constant number of additional tapes on which we count passes (but there is no space bound). Suppose that the total number of passes is at most $\log^k n$, for an arbitrary constant $k \in \mathbb{Z}^+$. By [HS08] (Lemma 4.8), $T_U \in \mathsf{DSPACE}(r^2(n) \log n) = \mathsf{DSPACE}(\log^{2k+1} n)$, which contradicts Fact 4.2.3 (3). □

Remark 4.2.4. This lemma refers to the (uniform) models of interest. We derive the lower bound (Section 4.3, Theorem 4.3.1) using Communication Complexity, which means that inherently this lower bound applies even to non-uniform machines. Hence, Lemma 4.2.2 leaves open the possibility that lower bounds on the non-uniform version of our model follow by lower bounds on the non-uniform version of $(r, s, 2)$-r/w stream algorithms. Even if this were the case then still it would have been impossible to prove *polynomial* lower bounds - as we do here for SMs. An $(O(\log n), s, 2)$-r/w stream algorithm can sort the input [GHS06]. It is easy to see that by sorting the indices of the input elements we can solve $\left(\mathrm{IP}_{2,N}^{\mathbb{F}}\right)^{\oplus,\pi}$, the problem for which we derive the lower bound on Stack Machines.

Comparison of proof techniques. Beame and Huynh-Ngoc [BHN08] apply Communication Complexity to obtain logarithmic lower bounds on the number of passes for approximating frequency moments. This work is superficially related to ours, since some of the notation, terminology and the general proof template inspired ours. In particular, both in our argument and [BHN08] there is a direct-sum construction to obtain a streaming lower bound as follows. Given the *actual* input instance the players *lift* it to many "decoy" instances (subinstances), they combine them appropriately, and then they simulate the machine on this extended input instance. We borrow this idea of mapping a small input instance to its lifted analog. The goal is to argue that a Stack Machine must dedicate its stack resource to solving only a fraction of all subinstances. Other than this superficial similarity to [BHN08, GHS06], the heart of our argument is fundamentally different. The unboundness of the stack and the fact that it is a special type of memory determines the "algorithmic" part of our proof which is a reduction. In this sense our simulation is entirely new. Without getting further into details, at an intuitive level it is sufficient to observe that our proof completely breaks down when instead of one we have two stacks - with two unbounded stacks we can compute everything in a single pass. On the other hand, the simulation in [BHN08] holds for any number of *passes-bounded* tapes.

4.3 Statement of the main theorem and proof outline

We show a lower bound for P-Eval, where we identify evaluating a polynomial, from a family of quadratic polynomials, by computing $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$. Recall the definition of $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi} : \{0,1\}^{2mN} \to \{0,1\}$, and let the length of the domain argument be $n = 2mN$.

Theorem 4.3.1 (Main Theorem (restated))**.** *Let $\mathbb{F}$ be any field and M be any (r,s)-SM computing $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$. Then for every $\epsilon > 0$, either $r = \omega(n^{1/4-\epsilon})$ or $s = \omega(n^{1/4-\epsilon})$. Moreover, this holds even for a randomized (r,s)-SM M with two-sided error $\delta < 1/2$.*

This theorem is a corollary of our main technical contribution which is a Communication Complexity direct-sum type of argument.

Theorem 4.3.2 (Main Reduction)**.** *Let $c > 0$. Assume there exists a randomized (r,s)-SM M for $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$ with at most error $\delta < 1/2$, for $\pi \in \mathsf{Sym}(m)$, $m = N^c$. Then, there exists a randomized protocol P for $\mathrm{IP}_{2,N}$ with cost $O(r(2mN) \cdot \log(2r(2mN)) \cdot s(2mN))$ and error at most $\delta + d(1-\delta)$.*

Recall that $d = O(r(2mN)\log(2r(2mN))/\sqrt{m})$, and by the choice of the parameters (functions r, s) in the proof of Theorem 4.3.1, d approaches 0 as N gets to infinity.

Overview of the proof. Here is a high-level description of the argument overall.

We show that evaluating a polynomial at a given point requires polynomially many passes on the input tape. For this we rely on the lower bounds for IP (see Section 4.1). Theorem 4.3.1 follows by adjusting the parameters in the statement of Theorem 4.3.2. The reduction of Theorem 4.3.2 is the whole argument, where we present a randomized protocol that efficiently simulates an (r,s)-Stack Machine.

Our goal is to construct a *randomized protocol* P which runs on the given input x. For clarity of presentation, as an intermediate step we construct a *deterministic protocol* P' where the players in P use P' as a routine. In particular, in the description of P the players first transform (i.e. lift, permute, and combine) the given actual input x into an artificial bigger input v, using (i) the actual input x and (ii) shared randomness. Then, they simulate the deterministic protocol P' on v. The extension of the given input is such that the output to this input is the same as for the actual one. Note that P is a randomized protocol which uses its randomness to construct a distribution of inputs for the deterministic protocol P'.

The construction of P' is the bulk of the argument. It operates in a way that the other technical parts of the argument can be put together. The intuitive goal is to show that in P' it is possible for the players to perform the simulation of the Stack Machine without communicating the stack content. At first, due to the unboundness of the stack, this seems unexpected. The

argument relies on the specifics of the stack memory. In Section 4.4 we give an example which may shed light into some of the issues.

Here are the main steps of the reduction (Theorem 4.3.2).

1. Identify an obstruction to efficient simulation in P'. This is the definition of a *corrupted instance* (Definition 4.4.3). The intention is that in absence of the obstruction it is possible to simulate the (r, s)-Stack Machine without communicating the stack content.

2. Bound the frequency of this obstruction (Lemma 4.4.6).

3. Construct the deterministic protocol P' (Lemma 4.4.5). This protocol may abort the simulation when we have a corrupted instance. However, the protocol P' detects when in v a corrupted instance occurs. When we do not have a corrupted instance then the protocol works correctly and it is efficient, in the sense that the players do not have to communicate the stack content during the simulation of the machine.

4. Use Yao's min-max principle to show the existence of a randomized protocol P (Theorem 4.3.2). This is just a technical twist that uses distributional complexity to take care of distribution-wise problems in the inputs of the constructed protocol.

4.4 Main lemmas, intuition, and examples

We introduce some additional terminology and notation. We define the concept of a corrupted instance, and introduce notation for the construction of the distribution of inputs x' in P when given the actual input x. We couple the definitions with intuitive remarks. Then, we state the lemmas that constitute the technical part of the reduction. We conclude this section by giving some examples providing insight to the proofs that follow. The actual arguments are given in the next Section 4.5.

4.4.1 More definitions and notation

Here is a list of the basic terminology and notational conventions. For convenience we also summarize the notation that was fixed earlier.

- We deal with two problems.

    1. The usual inner product $\mathrm{IP}_{2,N}$, on inputs of length $2N$. In the Communication Complexity model each player gets a vector of N bits.
    2. The other is $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$. This is the lifted version of $\mathrm{IP}_{2,N}$ on m *sub-instances*. We also refer to those as IP*-instances*, or simply as *instances* when it is clear from the

context. The number of IP-instances is determined by π in the notation $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$. Note that $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$ is a boolean function with domain $\{0,1\}^{2mN}$.

- Let $n = 2mN$. In this notation, $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$ has domain $\{0,1\}^n$.
- n, m, N are polynomially related.
- The permutation π (from a family of permutations, for infinitely many m's), is of sortedness $O(\sqrt{m})$.
- To reduce notational clutter in the subsequent definitions, we make the convention that in $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$ the index of the first part of the j-th subinstance is identified using the identity permutation $\pi_1 := \mathsf{id}$, whereas the second part of the j-th sub-instance with the permutation $\pi_2 := \pi$. In this notation, $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}(x) = \bigoplus_{j=1}^{m} \mathrm{IP}_{2,N}(x_{[j],\pi})$, where $x_{[j],\pi} = (x_{1,j}, x_{2,\pi_2(j)}) = (x_{1,\pi_1(j)}, x_{2,\pi_2(j)})$.
- Recall (see Section 3.1) that for a Stack Machine each transition exactly one of the following happens: either a head-move, or a push to the stack or a pop from the stack. We refer to *move transitions*, *push transitions* and *pop transitions*, respectively.

For a field $\mathbb{F}$ and $\mathrm{IP}^{\mathbb{F}}_{2,N}$ we write 0-*inputs* to refer to the subset of $\mathbb{F}^{2N}$ which consists of every element whose inner product is 0. This is not to be confused with the all-0 vector in $\mathbb{F}^{2N}$ which is just one 0-input.

Let us put in perspective the following Definition 4.4.1 by giving a preview of a part of protocol P. Given a Stack Machine M for $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$, we build a communication protocol for $\mathrm{IP}_{2,N}$. Let x be an input to P. Our strategy is as follows: the players in P extend x to an input v to $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$ by randomly choosing an instance number $j \in \{1, \ldots, m\}$, randomly choosing $m-1$ many *0-instances* $\overline{y} = (y^1, \ldots, y^{m-1})$ to IP, *embedding* x as instance number j and $\overline{y}$ as the other instances in an input $v(j, x, \overline{y})$ to M, and simulating M on input v.

Definition 4.4.1 (Embedded input instance)**.** Let $N, m \in \mathbb{Z}^+$ and $\pi \in \mathsf{Sym}(m)$. Also, let $j \in \{1, \ldots, m\}$, $x \in \{0,1\}^{2N}$ and $\overline{y} = (y^1, \ldots, y^{m-1}) \in \{0,1\}^{2(m-1)N}$. Recalling Definition 4.1.1, we define $v = v(j, x, \overline{y}) \in \{0,1\}^{2mN}$ to be the string satisfying: $v_{[j]} = x$; $v_{[j']} = y^{j'}$ for each $1 \le j' < j$; and $v_{[j']} = y^{j'-1}$ for $j < j' \le m$.

Since $\overline{y}$ will consist of 0-instances to $\mathrm{IP}_{2,N}$, it's clear that $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}(v(j, x, \overline{y})) = \mathrm{IP}_{2,N}(x)$. Thus, if the players in P successfully simulate M on $v(j, x, \overline{y})$, then they obtain $\mathrm{IP}(x)$.

Remark 4.4.2. Here is a remark and preview of P, regarding the relevance of the notation. Assume M is deterministic, and let Γ denote the sequence of configurations of M on input $v = v(j, x, \overline{y})$. The players in protocol P collectively see most of the tape of M, except for the

parts where their respective inputs are embedded into $v(j, x, \overline{y})$. Specifically, Player-$i \in \{1, 2\}$ is the only one who sees the symbols in $v_{i,\pi_i(j)}$, (recall that π_1 is the identity, $\pi_1(j) = j$) which is x_i. In protocol P, the players will take turns simulating the transitions in Γ one by one, and it will be the job of Player-i to simulate the transitions when the input head is scanning $v_{i,\pi_i(j)}$. Intuitively, we expect that by using $m > 1$ and the sequence of small sortedness induced by π, the machine M does not use its stack on instance j, which is the one we really want to solve.

Definition 4.4.3 (Corrupted instance)**.** Let $j' \in \{1, \ldots, m\}$ and let $v' \in \{0,1\}^{2mN}$, for any $m, N \in \mathbb{Z}^+$. [3] Let M' be a deterministic (r, s)-Stack Machine. Let Γ' be the sequence of configurations of M' on v'. Partition Γ' into r contiguous passes according to the movement of the input head. We say that *instance j' is corrupted in input v' on machine M'* if the following holds. There exist scans $l_1 \leq l_2 \in \{1, \ldots, r\}$, and let the two players identified by $i_1, i_2 \in \{1, 2\}$, $i_1 \neq i_2$, and configurations $\gamma_1, \gamma_2 \in \Gamma'$ such that:

(i) for $a \in \{1, 2\}$, γ_a belongs to scan l_a and the input head in γ_a is scanning a symbol from $v'_{i_a,\pi_{i_a}(j')}$

(ii) the transition out of γ_1 is a push transition

(iii) the transition out of γ_2 is a pop transition

(iv) the stack height is strictly higher between γ_1 and γ_2

(v) the in-between stack height is strictly higher

Let $\textsc{Bad}(M', v') \subseteq \{1, \ldots, m\}$ denote the set of instances that are corrupted in v' on M'.

Remark 4.4.4. In Figure 4.1 we depict a typical case. This diagram helps in the understanding of the subsequent ones. Recall that $v_{[j]} = (v_{1,j}, v_{2,\pi(j)})$ is an IP-instance. Think of $v_{1,j}$ to be known only to Player-1, whereas $v_{2,\pi(j)}$ to be known only to Player-2. If Player-1 pushes a stack symbol and this symbol is popped when the head of the machine is over a part of the tape that he has no knowledge about, then neither Player-1 can carry the simulation (he doesn't know the input) nor Player-2 (she doesn't know the stack symbol).

4.4.2 Statements of the main lemmas

There are two main lemmas which we state explicitly, and one more which is the application of Yao's min-max principle which is implicit in the proof of Theorem 4.3.2.

The bulk of the proof is the simulation when the actual input x (for $\mathrm{IP}_{2,N}$) is embedded in the lifted input for a position j which is *not* corrupted in $v(j, x, \overline{y})$ (this is an input for $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$) on M'. Then, the protocol P' efficiently simulates M'.

[3] In this definition, j' and v' are *not* necessarily related by $v' = v(j', x, \overline{y})$ for some $x, \overline{y}$.

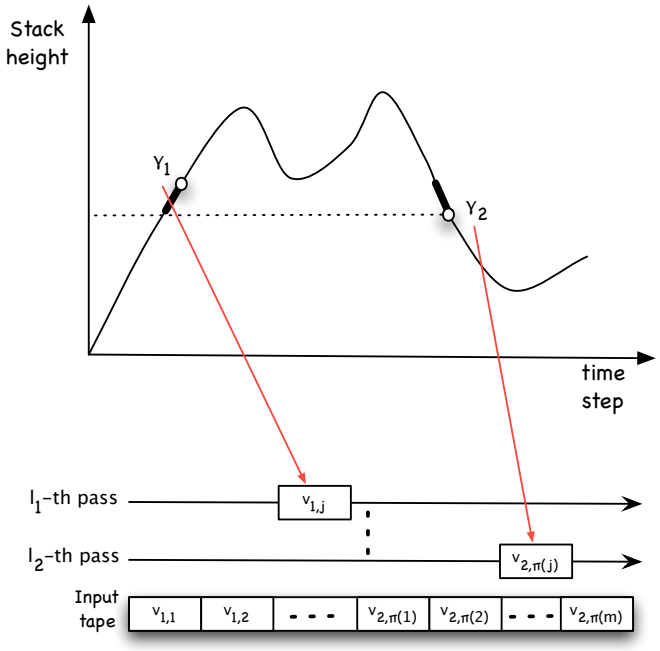


Figure 4.1: Configuration γ_1 is a push configuration, whereas configuration γ_2 is a pop configuration. In γ_1 the input head is over $v_{1,j}$ whereas in γ_2 the head is over $v_{2,\pi(j)}$.

Lemma 4.4.5 (Deterministic protocol - simulation)**.** *Let M' be a deterministic (r,s)-Stack Machine for $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$. Let $j \in \{1,\ldots,m\}$ and let $\overline{y} \in \{0,1\}^{2(m-1)N}$. There exists a deterministic communication protocol $P' = P'(M', j, \overline{y})$ such that, on input $x \in \{0,1\}^{2N}$:*

(i) if $j \in \textsc{Bad}(M', v(j,x,\overline{y}))$, then P' outputs "fail"

(ii) otherwise, P' outputs $M'(v(j,x,\overline{y}))$

and the cost of P' is $O(r(2mN)\cdot\log(2r(2mN))\cdot s(2mN))$, i.e. $O(r(n)\cdot\log(2r(n))\cdot s(n))$.

We would like to make sure that in protocol P, the probability that instance j is corrupted on $v(j,x,\overline{y})$ is small. In the following lemma, it is not hard to see that the small sortedness of π implies the stated upper bound on the number of instances that are corrupted in a fixed input v'.

Lemma 4.4.6. *Let M' be a deterministic (r,s)-Stack Machine for $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$ and let $v' \in \{0,1\}^{2mN}$. Then, $|\textsc{Bad}(M', v')| = O(r(2mN)\cdot\log(r(2mN))\cdot\sqrt{m})$.*

4.4.3 Intuitive remarks

We provide intuitive explanations and typical diagrams for some ideas in the arguments. The definitions are in Section 4.4.1, whereas the proofs are given later on in Section 4.5.

Frequency of the obstruction. In Lemma 4.4.6 we show an $O(r(n)\log r(n)\sqrt{m})$ lower bound on the number of bad corrupted j's (for fixed input and machine M). In fact, a lower bound $O(r^2(n)\sqrt{m})$ can be easily obtained. Note that even with this weaker bound our proof still goes through with slightly different parameters.

This is the point where the sortedness of π becomes relevant. The weaker bound can be directly derived by observing Figure 4.2.

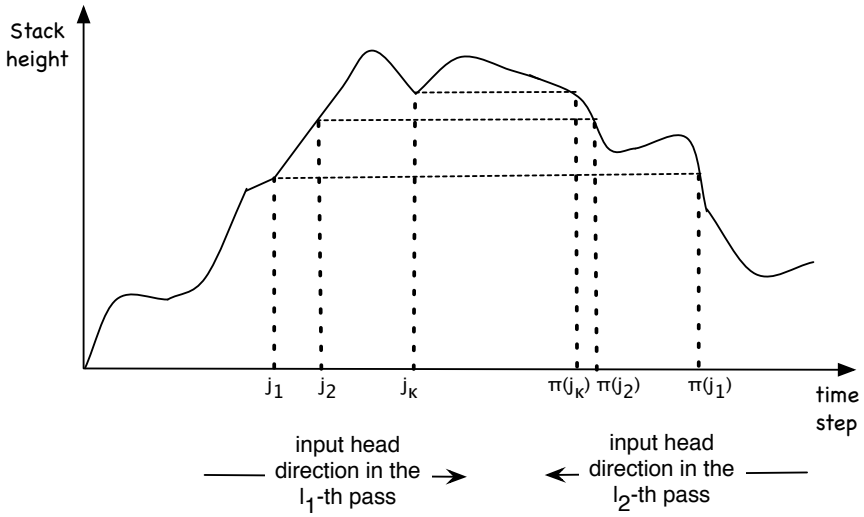


Figure 4.2: Fix an arbitrary pair of passes (l_1, l_2). Say that in l_1 the head moves left to right, and in l_2 it moves right to left. How many distinct j's can be corrupted because of (l_1, l_2)? Observe that π is such that the longest increasing sequence is of length $O(\sqrt{m})$.

Remark 4.4.7. If the head of the machine was not scanning the input from left to right and right to left but rather after making one pass it started over from the beginning in a left to right fashion, then permutations of small sortedness wouldn't be necessary for our argument[4]. Note that in this case we lose the immediate connection with work in Complexity Theory of AuxPDAs. This in particular implies that there is a problem which can be done with two passes when the head can reverse, but it requires $n^{\Omega(1)}$ many passes when we can scan the input only in a unidirectional fashion. In other words we obtain separation similar to [MMN10] (but we do it even for polynomially many passes).

Example for a special case of the efficient simulation. Consider the case of two players. Both players know the 0-inputs. We refer to the part of the 0-inputs as public input zone. If a stack symbol was pushed to the stack during the simulation of M' when the head was over a public zone then this stack symbol (for the corresponding time step and stack height) is a public

[4]I'd like to thank Anastasios Sidiropoulos for this observation.

stack symbol. This way we define the concept of public stack zone. Similarly, we define the private zones for Player-i. The symbols in the stack have value (i.e. the actual symbol) and in addition we label them according to which zone the input head was in when they were pushed in the stack. This distinction is necessary, because during the simulation, for a particular stack height a player may know the label of the input-zone associated with the stack symbol, but not the actual symbol.

Consider the situation depicted on Figure 4.3 and suppose that the simulation has reached time t_1. On the right we depict the stack content which was created by M by computing on the input and by possibly making a few scans over the tape between time 0 and t_1. Furthermore, assume the invariant that up to time t_1 each player knows: (i) the position (stack height) of every public and private stack-symbol, (ii) the symbol for each public stack symbol, (iii) Player-i knows her private stack symbols and (iv) the top stack symbol of the private stack zone which belongs to the other player. Player-1 will carry the simulation at this point since the input

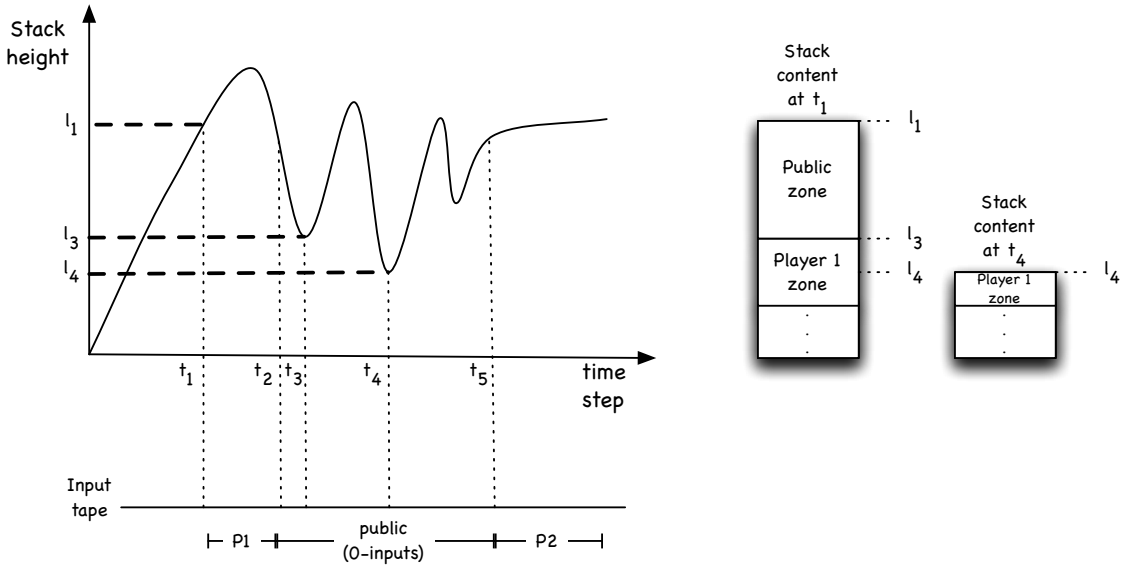


Figure 4.3: On the upper part we depict how the stack height changes as a function of the simulation step of M' on the given input. On the right side of this upper part we depict the stack content at two different times. On the lower part we depict the input and which part of the input is known only to player 1, only to player 2 and which is known to both (public).

head enters her private zone. She can do the simulation up to time t_2 where the head exits her private zone. As the simulation proceeds she pushes to the stack her private stack symbols and when she pops stack symbols these are either private to her or public. At time t_2 it is not a good idea to send to Player-2 the stack height and the input-head position. The reason is that later on and as long as the head is over the public zone, the popped stack symbols enter at

time t_3 the private zone of Player-1 (and Player-2 does not know these stack symbols). Thus, Player-1 continues to simulate after t_2 and up to the lowest level l_4 of the stack with the input head over the public zone. At this point and since she knows this is the lowest level, she sends to Player-2 the stack height, the top stack symbol, and the position of the input head. Since this is the lowest stack height and the input head is over the publicly known zone both Players can independently reconstruct the public stack zone from time t_4 all the way to time t_5, where Player-2 takes over the simulation.

4.5 Proofs of Theorems and Lemmas

4.5.1 Proof of Theorem 4.3.1

If $\epsilon \geq 1/4$ then the statement is vacuous. Let $\epsilon < 1/4$ and $\beta := 1/4 - \epsilon$. Since $0 < \beta < 1/4$ we have

$$0 < \frac{1}{1-2\beta} < \frac{1}{2\beta}$$

Define α such that $\alpha + 1$ is in between the two quantities above, i.e. let

$$\alpha := \frac{1}{2}\big(\frac{1}{1-2\beta} + \frac{1}{2\beta}\big) - 1$$

Then, $1/(1-2\beta) - 1 < \alpha$ and $\alpha < 1/(2\beta) - 1$. Hence,

$$\beta(\alpha+1) < \alpha/2$$
$$2\beta(\alpha+1) < 1$$

Let $m = N^\alpha$.

Assume that there exists randomized SM M that computes $\left(\mathrm{IP}^{\mathbb{F}}_{2,N}\right)^{\oplus,\pi}$ with error δ, and works in space $s = O(n^\beta)$ and makes at most $r = O(n^\beta)$ passes over the input tape. Then, by Lemma 4.4.5 there exists a randomized communication protocol for $\mathrm{IP}_{2,N}$ of cost $O(r\log(r)s)$ and error at most $\delta + d(1-\delta)$, where $d = d(N) = O(r(2mN)\log(r(2mN))/\sqrt{m})$ which for the chosen parameters $d \to 0$ as $N \to \infty$. Hence, for sufficiently large inputs the error of the protocol becomes $< 1/2$ and by standard arguments, repeating P a constant number of times and taking majority vote, we get a protocol for $\mathrm{IP}_{2,N}$ of cost $o(N)$ with error at most $1/3$, which contradicts [CKS03].

4.5.2 Proof of Theorem 4.3.2

Let M_q denote the deterministic (r,s)-Stack Machine obtained by running M with random string q.

Let D be any distribution on the space $\{0,1\}^{2N}$ of inputs to $f = \text{IP}$. Below, we give a randomized protocol P_D for f with cost and error as described in Theorem 4.3.2, but with error computed over the choice of the input x from D *and* of the random coins ρ; note that these random coins is the public randomness, and in particular q will be a substring of ρ. By standard arguments (due to the non-uniformity of the Communication Complexity model), we can fix ρ to obtain a deterministic protocol with the same error, but only over the choice of x from D. Since such a protocol exists for every distribution D, by Yao's min-max principle (e.g., Theorem 3.20 in [KN97]), the conclusion of Theorem 4.3.2 follows.

If D has support only on 0-inputs or only on 1-inputs, P_D is trivial. Otherwise, let $D_0 = D$ conditioned on $f(x) = 0$.

Consider the following protocol P_D:

On input $x \in \{0,1\}^{2N}$, where Player-i gets $x_i \in \{0,1\}^N$, and shared random string ρ:
Publicly draw j uniformly from $\{1,\ldots,m\}$
publicly draw $\overline{y} = (y^1,\ldots,y^{m-1})$ from D_0^{m-1}
publicly draw q uniformly
Run $P' = P'(M_q, j, \overline{y})$ (from Lemma 4.4.5) on input x, simulating $M_q(v(x, j, \overline{y}))$
If P' outputs "fail", then P_D outputs 1. Else, P_D outputs the same value as P'

The cost of P_D is equal to the cost of P' which is $O(r(2mN))s(2mN))$. To argue about the error in P_D we define the following events:

$A = A(x,\rho)$: P_D is correct on input x with coins ρ
$B = B(q,v)$: M is correct on input v with coins q and
$C = C(j,q,v)$: $j \notin \text{BAD}(M_q, v)$.

By Lemma 4.4.5, the simulation in P' fails if and only if $\overline{C}$. Let $\alpha = \Pr_x[f(x) = 0]$. We write $\Pr[A] = \alpha \cdot \Pr[A|f(x) = 0] + (1-\alpha) \cdot \Pr[A|f(x) = 1]$.

For the right term, consider the conditioning $f(x) = 1$. Then, P_D is correct if either the simulation in P' fails (and P_D correctly outputs 1), or the simulation in P' does not fail and M is correct, thus, $\overline{C} \vee (C \wedge B) \Rightarrow A$, in particular $B \Rightarrow A$, so $\Pr[A|f(x) = 1] \geq \Pr[B|f(x) = 1]$. The event $f(x) = 1$ is independent of the choice of q, and by the correctness condition of M, $\forall v, \Pr_q[B(q,v)] \geq (1-\delta)$. Hence, $\Pr[B|f(x) = 1] = \Pr[B] \geq (1-\delta)$.

For the left term, consider the conditioning $f(x) = 0$. Then, P_D is correct whenever the simulation in P' works and M is correct, thus $B \wedge C \Rightarrow A$, and $\Pr[A|f(x) = 0] \geq \Pr[B|f(x) = 0] \Pr[C|B, f(x) = 0]$. As before, $\Pr[B|f(x) = 0] \geq (1-\delta)$. Next, j is clearly statistically independent of x and q. Furthermore, under the conditioning $f(x) = 0$, we claim that j *is statistically independent from* v. To see this, notice how the distribution obtained

on pairs (j, v) in P_D constitute the same as independently choosing m 0-inputs from D_0 and combining them in v, and choosing j uniformly from $\{1, \ldots, m\}$. Thus, in the expression $\Pr[C(j, q, v)|B(q, v), f(x) = 0]$, the conditioning depends on (x, v, q), C itself depends on j, and j is statistically independent from (x, v, q). By Lemma 4.4.6, $\forall (q, v), \Pr_j[C(j, q, v)] \geq (1 - d)$. Then, $\Pr[C|B, f(x) = 0] = \Pr[C] \geq (1 - d)$.

Putting everything together gives the claimed error bound, $\Pr[A] \geq 1 - (\delta + d(1 - \delta))$.

4.5.3 Proof of Lemma 4.4.6

Let $i_1 \neq i_2 \in \{1, 2\}$ be the identifiers for the two players as in Definition 4.4.3. We say that *instance* $j \in \textsc{Bad}(M', v')$ *is corrupted by the pair* (l_1, l_2) if the latter is the lexicographically smallest such tuple satisfying the conditions in Definition 4.4.3 for instance j. We upper bound the number of instances corrupted by one pair.

Let $J \subseteq \{1, \ldots, m\}$ be the distinct instances corrupted by the same pair (l_1, l_2).

Claim 4.5.1. $k := |J| = O(\sqrt{m})$

Proof. Let these instances be indexed by $j_1, \ldots, j_k$ such that $\pi_{i_1}^{-1}(j_1) < \pi_{i_1}^{-1}(j_2) < \cdots < \pi_{i_1}^{-1}(j_k)$. Assume that both l_1 and l_2 are odd. Then, the head of M' scans v_{i_1} during pass l_1 from left to right. So, M' first visits $v_{i_1, \pi_{i_1}^{-1}(j_1)}$, then $v_{i_1, \pi_{i_1}^{-1}(j_2)}$, and so on, up to $v_{i_1, \pi_{i_1}^{-1}(j_k)}$. Recalling Definition 4.1.1, if $\pi_{i_1} = \pi$ (recall that $\mathsf{sortedness}(\pi) = O(\sqrt{m})$) then $k = O(\sqrt{m})$. Else, suppose that $\pi_{i_1} = \mathsf{id}$ and $\pi_{i_2} = \pi$. For $a \in \{1, \ldots, k\}$, let $\gamma_{a,1}, \gamma_{a,2}$ be the configurations mentioned in Definition 4.4.3. By this definition, the stack level cannot drop between $\gamma_{a,1}$ and $\gamma_{a,2}$, so then we must have $\gamma_{1,1} \prec \gamma_{2,1} \prec \cdots \prec \gamma_{k,1} \prec \gamma_{k,2} \prec \cdots \prec \gamma_{2,2} \prec \gamma_{1,2}$. Since we assumed l_2 is also odd, the head is also moving left to right in pass l_2, so we obtain that $\pi_{i_2}^{-1}(j_k) < \cdots < \pi_{i_2}^{-1}(j_2) < \pi_{i_2}^{-1}(j_1)$. Therefore, $k = O(\sqrt{m})$. But either $\pi_{i_1} = \pi$ or $\pi_{i_2} = \pi$ (and the other player has the identity permutation), where recall that $\mathsf{sortedness}(\pi) = O(\sqrt{m})$. It is easy to see that the relative parities of l_1 and l_2 change this argument only in that we might obtain a monotone increasing subsequence instead of a monotone decreasing one. Thus, every pair can corrupt at most $O(\sqrt{m})$ instances. □

Next, consider $A \in \{0, 1\}^{r \times r}$, where $A[l_1, l_2] = 1$ iff some instance is corrupted by the pair (l_1, l_2). We count the number of 1s in A. Note that A is upper triangular, because for a pair to corrupt an instance we must have $l_1 \leq l_2$.

Moreover, for $k \geq 1$, consider the diagonal $l_2 - l_1 = k$, and two entries on this diagonal that are k' cells apart, for $1 \leq k' < k$. We claim that we cannot have $A[l_1, l_1 + k] = 1$ and $A[l_1 + k', l_1 + k + k'] = 1$. Assume this were true. Then, some instance j is corrupted because a symbol is pushed on the stack in scan l_1 and popped in scan $l_1 + k$, and another instance j' is

corrupted because a symbol is pushed on the stack in scan l_1+k' and popped in scan l_1+k+k'. But, with these settings, $l_1 < l_1 + k' < l_1 + k < l_1 + k + k'$, which contradicts the way a stack works.

Hence, for $k \geq 1$, the diagonal $l_2 - l_1 = k$ can contain at most r/k 1s. In total, we have that A contains at most $O(r \log r)$ 1s. Thus, $|\textsc{Bad}(M', v')| \leq O(r \log r \sqrt{m})$.

4.5.4 Proof of Lemma 4.4.5

Let n, m, N, s, r be as in the lemma. Let M' be a deterministic nonuniform SM with space bound s and pass bound r. Let $j \in [m]$ and let $\bar{y}$ be a sequence of $m-1$ inputs to f.

Our goal is to show that there exists a deterministic 2-player communication protocol $P' = P'(M', j, \bar{y})$ that, on input x,

1. if $j \in \textsc{Bad}(M', v(j, x, \bar{y}))$, then P' outputs "fail";
2. otherwise, P' correctly simulates M' and outputs $M'(v(j, x, \bar{y}))$;
3. the cost of P' is $O(r \cdot \log(2r) \cdot s)$.

Fix an input x and let $v := v(j, x, \bar{y})$. Observe that the players in P' share j and $\bar{y}$, and their private inputs are x_1, x_2. Thus, from the input v to M', they each know all instances $i \neq j$, and from instance j, player p only knows x_p. Furthermore, observe that the goal of P' is *not* to compute f, but rather, to compute the output of M' on v.

Private Input Symbols. We say that *an input symbol* from v *is private to player* p if it's part of $v_{p,\varphi_p^{-1}(j)}$, which is where x_p, the input to player p, is embedded in v. Input symbols that are not private to any player are *public*.

Private Stack Symbols. Let Γ be the sequence of configurations of M' on v. Consider a configuration $\gamma \in \Gamma$ and look at the contents of the stack in γ. For every symbol ξ appearing on the stack, we say that *the stack symbol* ξ *is private to player* p if the input head was scanning an input symbol private to player p in the configuration prior to the transition in which ξ was pushed on the stack. A stack symbol that is not private to any player is *public*.

Input- and Stack-Private Configurations. We say that *configuration* γ *is input-private to player* p if the input head in γ is scanning an input symbol that is private to player p. We say that *configuration* γ *is stack-private to player* p if it is not input-private to player p, the transition out of γ is a pop transition and the top stack symbol in γ is private to player p. A configuration that is neither input- nor stack-private is *public*.

Intuitively, player p is responsible for simulating the transitions out of configurations that are input- or stack-private to itself. Note that, there exists a configuration that is input-private and stack-private to two different players if and only if $j \in \textsc{Bad}(M', v)$.

Hollow View. We say that *player p sees a hollow view of the stack* in a configuration γ if player p knows:

(i) the stack height;

(ii) for every symbol on the stack, whether it is public or private

(iii) all stack symbols that are public or private to itself; and

(iv) for $p' \neq p$, the top stack symbol in any contiguous zone of symbols private to player p'.

We say that *a player sees a hollow view of the configuration γ* if it knows the state of the SM, the location of the input head, and it also sees a hollow view of the stack in γ.

Some Simple Facts. In the protocol P', the players simulate Γ transition by transition, always using hollow views of the configurations along the way. We observe the following facts.

(1) If a player p sees a hollow view of a configuration γ, then that player can determine whether γ is public, input-private to the other player (since it knows the location of the input head), or stack-private. There is a mild subtlety involving the latter, specifically, the definition of γ being stack-private involves knowing that the transition out of γ is a pop transition. To this end, observe that, by virtue of part (iv) of the hollow view of the stack, if γ is not input-private, then player p knows the transition out of γ, even in the case when the top stack symbol is private to player p'.

(2) If a player sees a hollow view of a *public* configuration γ, then that player can compute a hollow view of the configuration following γ (denoted by $\mathsf{next}(\gamma)$).

(3) As long as $j \notin \textsc{Bad}(M', v)$, if player p sees a hollow view of a configuration γ which is input- or stack-private to player p, then player p can compute a hollow view of $\mathsf{next}(\gamma)$.

(4) If player p sees a hollow view of a configuration γ that is input-private to itself, it can detect whether γ is stack-private to p'.

The Protocol. We now describe the protocol P'. The sequence Γ is split into several contiguous sections, each of which is of exactly one of the following three types:

(a) *Public sections.* These sections start with *some* public configuration (not just any), they consist exclusively of public configurations, and they extend up to, and including, the first configuration which is no longer public.

Inductively, by facts (1) and (2) above, if both players have a hollow view of the configuration at the beginning of a public section, they can all simulate the entire public section starting at that configuration. At the end of the section, the control of the simulation is given to the other player. All this is done without any communication.

(b) *Input-private sections.* Each such section starts with, and contains, a maximal sequence of input-private configurations to some player p, and it ends with the single configuration immediately following that sequence. The latter might be public, or input-private to p', or stack-private to any of the players.

Inductively, by facts (1), (3) and (4) above, if player p sees a hollow view of the configuration at the beginning of a sequence of input-private configurations to itself, it can either detect that instance j is corrupted, in which case player p aborts the simulation and the protocol P' outputs "fail", or simulate the entire section. Let γ' be the configuration at the end of the section. At this point, player p communicates:

(v) the state in γ';

(vi) the input head position in γ';

(vii) the stack height L' in γ';

(viii) the top stack symbol in γ';

(ix) the lowest stack height L'' achieved in a configuration between γ and γ'; and

(x) the stack symbol at level L''.

(c) *Mixed stack-private and public sections.* Each such section begins with a configuration that is stack-private to a player p, and contains a mix of configurations that are either stack-private to p or public.

Let γ be the configuration at the beginning of such a section. Let p be the player to which it is stack-private, and assume this player has a hollow view of γ. Let γ' be the first configuration following γ which is either input-private to any player, or stack-private to p'. Inductively, by facts (1), (2) and (3) above, player p can simulate the entire sequence of configurations between γ and γ'.

The key technical point of this entire proof is determining the right place where a player stops a privately simulated section of type (c). It turns out that this is not γ': doing so could result in public symbols being placed on the stack which would only be known to

player p, unless they would be subsequently communicated, which would affect the cost of the protocol. Furthermore, it turns out it is also a bad idea for player p to end this section at the first public configuration encountered: the subtlety here is that the stack level might go up and down, and each time it goes down a few private stack symbols are being popped from the same contiguous section of private stack symbols. This scenario would hurt the cost of the protocol, for we could only bound the amount of communication in terms of the *length* of the contiguous zones of private stack symbols, which can be large.

Instead, player p, looking at the entire sequence from γ to γ', computes *the minimum possible stack level* for this section, and finds the last configuration γ'' preceding, and possibly equal to, γ' where the stack level is minimal. At this point, player p communicates (v)—(viii) for configuration γ''. Observe that, by definition of γ'', (ix) and (x) are equal to (vii) and (viii), respectively.

Having defined the types of simulated sections, we show that the players have enough information to carry out the simulation. Let A be the number of different sections in the simulation. We claim that for every $1 \leq a \leq A$, *all* players eventually see a hollow view of the configuration at the beginning of section a. We prove this by induction on a.

For $a = 1$, observe that the initial configuration is either public or input-private to player 1. In either case, all players see a hollow view of the stack, because the stack is empty.

Inductively, assume both players have a hollow view of the configuration γ at the beginning of section $a \geq 1$.

If the section is of type (a), by fact (2), all players see a hollow view of the configuration at the end of this section.

If the section is of type (b), let p be the player to which γ is input-private. Clearly, by fact (3), p itself sees a hollow view of the configuration γ' at the end of the section. After p communicates (v)—(x), player p' updates its view of the stack as follows: it truncates the stack at height L''; fills it to the height L' with symbols private to player p; to compute part (iv) of the hollow view, player p' obtains the top stack symbol in γ' from (ix), and the stack symbol at level L'' from (x). Taking into account (v) and (vi), now player p' sees a hollow view of γ'.

If the section is of type (c), let p be the player to which γ is stack-private. Let γ'' be the configuration where this section ends. Clearly, by facts (2) and (3), player p sees a hollow view of γ''. After p communicates (v)—(viii) at the end of the section, player p' updates its view of the stack as follows: it truncates the stack at height L'; and it updates the top stack symbol using (vii). Taking into account (v) and (vi), now p' sees a hollow view of γ''.

This completes the description of the protocol P'. If $j \notin \textsc{Bad}(M', v)$, the players get to the final configuration, obtaining the output of M' on input v. If $j \in \textsc{Bad}(M', v)$, one of the

players detects this during a section of type (b), the simulation is aborted, and P' and outputs "fail".

The Cost of the Protocol. We now compute the amount of communication in P'. The number of sections of type (b) is $2r$, one for each player, in every pass. For $1 \leq a \leq 2r$, let S_a be the a-th section of type (b) and let γ_a and γ'_a be the configurations at the beginning and at the end of S_a.

Fix $a < 2r$, and look at the stack in γ'_a. It contains public stack symbols, and several contiguous zones of private stack symbols, at most one such zone for every previous section of type (b). Furthermore, and crucially, observe that:

Claim 4.5.2. *The number of sections of type (c) in between S_a and S_{a+1}* equals *the number of different contiguous zones of private stack symbols from γ'_a (the end of S_a) from which a symbol is popped before γ_{a+1} (the beginning of S_{a+1}).*

Proof of Claim 4.5.2. This critical property follows from the way sections of type (c) are terminated. Specifically, we claim it is impossible for two different sections of type (c) between S_a and S_{a+1} to pop symbols from the *same contiguous zone* of stack symbols private to some player p.

Assume this was the case for some sections S'_1 and S'_2. In between them, there can be no input-private configurations (by definition of S_a and S_{a+1}), and also no configurations stack-private to player p' (otherwise, the zones of private stack symbols corresponding to S'_1 and S'_2 would not be contiguous). Hence, all configurations between S'_1 and S'_2 are either public, or stack-private to player p. Moreover, the stack level in S'_2 is strictly lower than in S'_1. Then, player p should not have ended section S'_1 so early, instead, it should have simulated the entire section of configurations between S'_1 and S'_2, and then it should have continued with S'_2.

Clearly, for every contiguous zone of private stack symbols from which symbols are popped in between S_a and S_{a+1}, there must be at least one associated section of type (c). By the argument above, there will be exactly one such section of type (c). □

Let γ_{2r+1} denote the final configuration. Consider the $(2r)\times(2r)$ matrix B, where $B[a_1, a_2] :=$ 1 if, in between γ'_{a_2} (the end of S_{a_2}) and γ_{a_2+1} (the beginning of S_{a_2+1} or the final configuration), a private stack symbol is popped that was pushed in S_{a_1}; and $B[a_1, a_2] := 0$ otherwise.

Claim 4.5.3. *There are at most $O(2r \cdot \log(2r))$ 1 entries in B.*

Proof of Claim 4.5.3. The matrix is 0 under the main diagonal, because for a symbol from S_{a_1} to be popped after S_{a_2}, we must have $a_1 \leq a_2$. Consider the diagonal $a_2 - a_1 = j$ for some $j \geq 1$. We claim that it is impossible to have $B[a, a + j] = 1$ and $B[a + j', a + j + j'] = 1$

for some a and some $0 < j' \leq j$. Assume $B[a, a+j] = 1$, so in between γ'_{a+j} and γ_{a+j+1} a symbol is popped that was pushed in S_a. But then, the stack in γ_{a+j+1} can no longer contain any symbols pushed in $S_{a'}$, for any a' such that $a < a' \leq a + j$. In particular, for $a' = a + j'$, all symbols pushed in $S_{a+j'}$ are no longer on the stack in γ_{a+j+1}, which precedes, or equals, $\gamma_{a+j+j'}$, Hence, it is impossible to have $B[a + j', a + j + j'] = 1$.

The argument above shows that the matrix B contains at most $2r/(j + 1)$ 1 entries on the diagonal $a_2 - a_1 = j$. Hence, in total, it contains at most $O(2r \log(2r))$ 1 entries. □

Lastly, we observe the following.

Claim 4.5.4. *The number of sections of type (c) in between S_a and S_{a+1} is at most the number of 1 entries in column a of matrix B.*

Proof of Claim 4.5.4. By Claim 4.5.2, every section of type (c) in between S_a and S_{a+1} is associated with a contiguous zone of stack symbols from γ'_a (the end of S_a) that are private to some player p. Since all private stack symbols pushed in $S_{a'}$ are contiguous on the stack (this is true at any point of the simulation), we see that there can be at most one section of type (c) for every section $S_{a'}$ such that symbols from $S_{a'}$ are popped in between S_a and S_{a+1}, or, equivalently, such that $B[a', a] = 1$. □

Putting everything together, first observe that sections of type (a) do not result in any communication. There are $2r$ sections of type (b), and by Claims 4.5.3 and 4.5.4, there are a total of $O(2r \cdot \log(2r)$ sections of type (c).

And the end of every section of type (b) or (c), items (v)—(x) are communicated. The full state takes $O(s)$ bits, the input head position takes $\log N \leq O(s)$ bits, the stack height takes $O(s)$ bits by Fact 3.1.3. Thus, every communication consists of $O(s)$ bits. The total cost of the protocol P' is thus $O(2r \log(2r) \cdot s)$.

Chapter 5

Some Remarks on Cryptography in Streaming Models

We initiate the study of Streaming Models for Private-Key Cryptography from generic cryptographic assumptions. This is the last, smallest, and technically simplest part of the thesis. The main technical contribution regards lower bounds. We also observe that it is possible to have logspace streaming Cryptography by directly applying, in a non-black-box way[1], the main construction of Applebaum, Ishai and Kushilevitz [AIK06a] on Cryptography in NC^0. The lower bounds we have obtained indicate that the connection with Cryptography in NC^0 is in some sense necessary.

In Section 5.1 we provide a preliminary motivating discussion which also improves the main result of [KGY89]. Basic definitions, notation, preliminary results and the non-black-box construction are given in Section 5.2. Note that there are certain changes with respect to the conventions made in the previous chapters. Section 5.3 investigates the possibility of basing the streaming computation of OWFs on known assumptions about OWFs, e.g. the hardness of Factoring or Subset-Sum, or on other generic assumptions. We answer these questions (unconditionally) in the negative by using standard tools from Communication Complexity, ideas from crossing sequence arguments, and elementary Information Theory techniques. In Section 5.4 we briefly discuss at an intuitive level the relation between graph-theoretic properties and cryptographic hardness. In Section 5.5 we provide slight improvements on the streaming computation of the OWF, and conjectures regarding logspace streaming cryptography. These conjectures shape the main theoretical motivation for studying Cryptography in Streaming models. [2] Finally, Section 5.6 concludes with our view on the future of Streaming Models for

[1] As usual, non-black-box means that the construction makes use of the "code" for computing the primitive.

[2] Also, in [Pap10] we have used the logspace streaming computation of the OWF to show under generic cryptographic assumptions that there is $\epsilon > 0$ such that $\mathsf{SAT}_{\mathsf{pw}}[\log^{\epsilon} n] \in \mathsf{PolyLogSpace} \cap \mathsf{NP}$ and $\mathsf{SAT}_{\mathsf{pw}}[\log^{\epsilon} n] \notin$

Cryptography.

5.1 Motivating discussion

Assumption 1 (p. 17), regarding the existence of 2^{n^ϵ}-hard OWFs, is the main assumption on which our constructions rely.

Example 5.1.1. We motivate the discussion by using this generic assumption to construct a OWF in a streaming manner. Let f be as in Assumption 1 computable in time and in particular in space n^k, for some constant k. Let $s(n) = \log^{k'} n$ for some $k' > k/\epsilon$. Consider an $O(s(n))$-bounded transducer which makes a single pass over the input and works as follows. Read the first $s'(n) := s(n)^{1/k}$ bits and copy them on the work-tape. Let this part of the input be $\alpha \in \{0,1\}^{s'(n)}$. Now, compute and output $f(\alpha)$. By the choice of k' we have that $2^{s'(n)^\epsilon} > p(n)$ for every polynomial function $p(n)$, and thus we obtain the following fact.

Fact 5.1.2. *If Assumption 1 holds, then there exists a OWF computable by a space bounded Turing Machine with a single pass over the input seed, for sufficiently large $\omega(\log n)$ space.*

In this example, it is crucial that $s'(n)$ is super-logarithmic. By strengthening Assumption 1 to hold for every $\epsilon < 1$ and requiring f to be computable in linear space, we construct a OWF for every $s(n) = \log^{1+\epsilon'} n$. But, when the workspace of the machine becomes $O(\log n)$ a similar construction is no longer secure. □

The above example can be extended to construct a PRG, if the OWF in Assumption 1 is a OWP. For this we apply the Goldreich-Levin construction (see Section 1.1.2) by partitioning $\alpha = (x, r)$ to output: $x, f(r), \langle x, r\rangle$, and we append to the output the remaining (i.e. excluding α) part of the seed.

Remark 5.1.3. Note that generalizing the above construction using (1.1.1), we can compute in polylogarithmic space a PRG of polynomial length. One question is whether the 2^{n^ϵ} hardness in the OWP is necessary for the construction of the PRG. The black-box separation in Section 2.3 implies that for natural variants of the above construction, we cannot get around the 2^{n^ϵ} hardness.

The following lemma improves on the main result[3] of [KGY89] and indicates the conceptual difficulty in constructing $O(\log n)$-streaming OWFs. Allender in his PhD thesis[4] [All85] obtained an even stronger result, which states that functions computable in the same way as in Lemma 5.1.4, are in fact invertible in NC.

BPP.

[3] [KGY89] shows that no PRG of linear stretch can be computed in logspace and a constant number of passes. Lemma 5.1.4 implies that no $\{0,1\}^n \to \{0,1\}^{n+1}$ PRG exists, since a PRG is in particular a OWF.

[4] I'd like to thank Eric Allender for bringing this result to my attention.

Lemma 5.1.4. *Let $k \in \mathbb{Z}^+$ be a constant, and f be a function computed by a logspace transducer that makes k passes over the input. Then, there exists an algorithm that given $f(x)$ computes $x' \in f^{-1}(f(x))$ in time polynomial in $|x|$.*
In particular, there does not exist a OWF computable in logspace with a constant number of passes over the input.

Proof. We rely on Corollary 3.5.11 which states that $\mathsf{NL}[O(1)] = \mathsf{NL}$. For given $y = f(x)$ consider the Turing Machine M that computes f and construct non-deterministic M' which guesses x and makes at most a constant number of passes over the non-deterministic tape and compares the outputted bits with y. Simulate, M' in polynomial time and output the certificate x for an accepting branch of the computation tree. □

5.2 Definitions and the logspace streaming computable OWF

5.2.1 Definitions and conventions

In this chapter we adopt, modulo a few modifications (see below), the definitions and notation of Section 3.1. For definitions of width parameters see Section 3.5.2. For definitions of Communication Complexity problems and results from previous work see Section 4.1.

For the space bound $s(n)$ of Turing Machines, we have $s(n) = \Omega(\log n)$. We parametrize on the number of passes over the input. Following the conventions in [AIK06a], we abuse notation and **we write NC^0 to refer to the functional analog of NC^0, where the circuits are assumed to be polytime uniform**[5]**.** All space-bounded TMs have an additional read-only, polynomially long advice tape which contains a P-uniform advice, unless mentioned otherwise. An $s(n)$ *space bounded transducer*, is a Turing Machine with a P-uniform advice tape, a read-only input tape, working memory of total size $s(n)$, and a write-only output tape. We abuse standard notation and we write $\mathsf{L/poly}$ to denote the class of functions computable by logspace transducers with a P-uniform advice. We define the class of functions $\mathsf{PASSES\text{-}SPACE}(r(n), s(n))$ to be the class of functions computable by $O(s(n))$ space bounded transducers that make at most $r(n)$ passes (i.e $r(n) - 1$ head-reversals) over the input.

The term $s(n)$-*streaming computation* refers to an $s(n)$-space transducer which makes $\log^{O(1)} n$ passes over the input. In general, *streaming computation* refers to $O(\log n)$-streaming computation.

Let $f \in \mathsf{NC}^0$ and let $\mathcal{C} := \{C_n\}$ be a family of NC^0 circuits computing f. We define the *family of dependency graphs* $\{G_n\}$ *of* $\mathcal{C}$ as follows. The *dependency graph* $G_n = ((I, O), E)$ *for*

[5]It is possible to weaken the form of uniformity to logspace uniform by additional assumptions for lattice problems.

C_n is a bipartite graph where I and O are the two sides of bipartization such that I, O are the set of input and output gates of G_n respectively. There exists an edge between $u \in I$ and $v \in O$ if there is a dipath of wires in C_n with endpoints u and v. If the circuit family is clear from the context then we write "dependency graphs of f" to refer to the dependency graphs of the particular circuits[6].

In the proof of Lemma 5.3.4 (see below) we make use of bipartite expanders. A *bipartite* (n, d, ϵ) *expander* is a graph $G = ((L, R), E)$, where $|L| = |R| = n$, it has a maximum degree d, and the following holds: for all $X \subseteq L$ where $|X| \leq n/2$, then $\Gamma(X) \geq (1 + \epsilon)|X|$, where $\Gamma(X)$ denotes the neighborhood of X. For an excellent overview, existence, and constructions of expanders see [HLW06].

5.2.2 Permuted inputs and outputs

The following simple observation is essential for the efficient streaming computation of OWFs. Consider permutations acting on strings as follows.

Definition 5.2.1. Let $x := x_1 \dots x_n \in \{0,1\}^n$ and $\pi \in \mathsf{Sym}(n)$, where $\mathsf{Sym}(n)$ is the group of permutations on $\{1, \dots, n\}$. Then, $\pi(x)$ denotes the string $x_{\pi(1)} \dots x_{\pi(n)}$. A family of permutations $\{\pi_n\}$ is $t(n)$-*time index-computable* if given n and j the value $\pi_n(j)$ is computable in time $O(t(n))$.

The security of OWFs is invariant under permutations acting on domain or image elements.

Fact 5.2.2. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a OWF. Then, for every x and polytime index-computable π on $|x|$ letters, and τ on $|f(x)|$ letters, the functions $f(\pi(x))$, $\tau(f(x))$ and $\tau(f(\pi(x)))$ are also OWFs.*

Similarly to OWFs the above fact holds for PRGs.

5.2.3 A OWF in PASSES-SPACE$(\log^{O(1)} n, \log n)$

The next Section 5.3 indicates the conceptual difficulty of constructing a OWF in logarithmic space. Indeed, we believe that the straightforward construction we present in this section, wouldn't be possible without (e.g. reproving parts of) the work for Cryptography in NC^0. We will just be using the statement of the theorem from [AIK06a] - see [AIK08] and also [AIK06b, AIK06a, AIK07] for the quite interesting details.

In [AIK06a] the authors show how to compile any function f computable in polynomial time as a function $f' \in \mathsf{NC}^0$, where f' in some sense preserves security properties of f. For this

[6] One could have similarly defined the dependency graph of the function f, but in this work the relevant concepts are about families of circuits under certain transformations.

compilation they need one more assumption, namely that there exists an Easy-PRG, i.e. a PRG computable[7] in $\oplus\mathsf{L}/\mathsf{poly}$ that stretches n bits to $n+1$. In other words, a polytime computable function can be compiled as a function in NC^0 preserving security properties, given that a PRG in $\oplus\mathsf{L}/\mathsf{poly}$ exists. Using the recent work of Haitner, Reingold, and Vadhan [HRV10] this assumption can be weakened to an Easy-OWF.

Assumption 3. There exists an Easy-OWF (OWF), which is a OWF computable in $\oplus\mathsf{L}/\mathsf{poly}$.

Therefore, a corollary of [AIK06a, HRV10] is the following theorem. Given a very hard OWF f there is $\epsilon > 0$ such that $F(xx') = f'(x)00\ldots0$ is one-way, where f' is the NC^0 analog of f and $|x| = \log^{\epsilon} n$, $n = |xx'|$. Then, we compute in logspace and poly-log many passes by computing each (among the poly-log many) output bits of $f'(x)$. This construction is straightforward but still non-black-box - we really need the description of the NC^0 circuit that computes the OWF to do the logspace computation.

Theorem 5.2.3. *Suppose that Assumption 1 and Assumption 3 hold true; i.e. a $2^{n^{\epsilon}}$ - hard OWF and an EOWF exist. Then, there exists $\epsilon' > 0$ and a OWF $F \in$ PASSES-SPACE$(\log^{\epsilon'} n, \log n)$.*

Given the discussion in Example 5.1.1 it seems counter intuitive why such a construction is at all possible. This is achieved using the machinery of Cryptography in NC^0. Some improvements are also possible - see Section 5.5. Moreover, the lemmas in Section 5.5 relate to our main conjectures relating Cryptography in Streaming Models with fundamental questions in Cryptography.

5.3 Obstacles to Streaming Cryptography - On the necessity of Cryptography in NC^0

Recall the construction of Example 5.1.1, and the lower bound of Lemma 5.1.4. Intuitively, starting from an *arbitrary* very hard OWF π it seems impossible to reduce the working space of the machine to $O(\log n)$ and still be able to securely compute a OWF π' without making too many passes over the input seed. In this section we discuss in greater detail why we cannot base Streaming Cryptography on standard and other plausible assumptions, *without making additional assumptions regarding cryptographic hardness, or the existence of cryptographic functions with special properties of their computation.* In other words, we provide conceptual evidence that goes against Theorem 5.2.3.

We show (i) that computing instances for Factoring and Subset-Sum by adding and multiplying numbers requires simultaneously polynomial space and passes over the input seed,

[7] Recall that P-uniform $\oplus\mathsf{L}$ is the class of languages characterized by non-deterministic logspace machines which accept iff the number of accepting paths is odd. The functional $\oplus\mathsf{L}/\mathsf{poly}$ is defined similar to L/poly.

(ii) in general, there is no simulation between NC^0-circuits (as always our circuits have multiple outputs) and logspace transducers with polynomially many passes over the input, and (iii) in general, there is no simulation of a $O(\log^{1+\epsilon} n)$-streaming computation by a $O(\log n)$-space bounded TM that makes $n^{\epsilon'}$ many passes over the input, for some $\epsilon' > 0$.

5.3.1 Impossibility of constructions based on Factoring or Subset-Sum

We show that computing the addition and the multiplication of two numbers requires simultaneously polynomial number of passes and polylogarithmic space. Let Add, Mult denote the respective computational problems.

Recall, Lemma 4.2.1 which states that an $s(n)$-space bounded TM for $\mathrm{IP}_{2,n}$ or $\mathrm{SetInt}_{2,n}$ requires $\Omega(\frac{n}{s(n)})$ passes. Instead of obtaining the lower bound for Mult and Add by modifying the reduction in the proof of Lemma 4.2.1, we provide a unified treatment.

Definition 5.3.1. Let $f, g : \{0,1\}^* \to \{0,1\}^*$. We write $f \leq_{c,l} g$, *f reduces to g via a $(c(n), l(n))$ passes-preserving logspace reduction*, where l is injective, if there exist logspace transducers M, M', such that for every string x, $|M(x)| = l(|x|)$, M makes at most $c(|x|)$ passes, and $f(x) = M'\big(g(M(x))\big)$, where M' makes a single pass over its input.

Typically, we are interested in passes-preserving reductions where $c(n)$ is a small constant (usually one or two). The above definition can be generalized by adding as parameter the number of passes of M' (not necessary generality for our results). The following is immediate by definition.

Fact 5.3.2. *Suppose that $f \notin \mathsf{PASSES\text{-}SPACE}(r(n), s(n))$, $s(n) = \Omega(\log n)$, such that $f \leq_{c,l} g$. Then, g cannot be computed with $l^{-1}(\frac{r(n)}{c(n)})$ passes in space $s(n)$.*

Lemma 5.3.3. Mult *cannot be computed in $o(\sqrt{\frac{n}{s(n)}})$ reversals in space $s(n)$, and* Add *cannot be computed in $o(\frac{n}{s(n)})$ reversals in space $s(n)$.*

Proof. It suffices to show that $\mathrm{IP} \leq_{2,n^2}$ Mult, and that $\mathrm{SetInt} \leq_{1,2n}$ Add.

$\mathrm{IP} \leq_{2,n^2}$ Mult: Let $\alpha = (a_1, \ldots, a_n), \beta = (b_1, \ldots, b_n) \in \{0,1\}^n$ be an instance of IP. Then, in two passes we construct (in logspace) the input

$$\alpha' = (a_1, \underbrace{0, 0, \ldots, 0}_{n\text{times}}, a_2, \ldots, a_{n-1}, \underbrace{0, 0, \ldots, 0}_{n\text{times}}, a_n), \quad \beta' = (b_n, \underbrace{0, 0, \ldots, 0}_{n\text{times}}, b_{n-1}, \ldots, b_2, \underbrace{0, 0, \ldots, 0}_{n\text{times}}, b_1)$$

where α', β' are the binary representations of two integers. In the integer multiplication of $\alpha'\beta'$ the $2n$-th bit corresponds to the first, least significant, bit of the sum $a_1b_1 + \ldots a_nb_n$, which coincides with the valuation of this sum over $\mathsf{GF}(2)$.

SetInt $\leq_{1,2n}$ Add: Let $x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n) \in \{0,1\}$, be the characteristic vectors of two sets over a universe of n elements. In a single pass construct $x' = (x_1, 0, x_2, 0, \ldots, 0, x_n), y' = (y_1, 0, y_2, 0, \ldots, 0, y_n)$. The intercalate zeros are carry-detectors. By adding $x + y$ there exists a carry-detector equal to 1 iff $\mathsf{supp}(x) \cap \mathsf{supp}(y) \neq \emptyset$. □

In particular, this implies that Mult, Add $\notin$ PASSES-SPACE$(n^{\epsilon}, \log^{O(1)} n)$, for some $\epsilon > 0$.

Permuted inputs. It becomes more involved, and we omit presenting, lower bounds for computing instances for Factoring and Subset-Sum when the inputs are permuted. Although for Mult a lower bound for permuted inputs can be obtained, the same is not true for Add. We have to consider the actual instance to Subset-Sum, which consists of $n^{\Omega(1)}$-many n-bit numbers, together with the characteristic vector of the set of numbers to be added.

5.3.2 Incomparability of NC^0 and PASSES-SPACE$(n^{\Omega(1)}, \log n)$

Recall that the decisional $\mathsf{NC}^0 \subsetneq$ Decision-PASSES-SPACE(O(1),O(1)) =: REGULAR. We show that the functional NC^0 and PASSES-SPACE$(n^{\hat{\epsilon}}, \log n)$ are incomparable in some strong sense, for some constant $\hat{\epsilon} > 0$.

Lemma 5.3.4. *(i) PASSES-SPACE$(1,1) - \mathsf{NC}^0 \neq \emptyset$, (ii) NC^0 − PASSES-SPACE$(n^{\hat{\epsilon}}, \log n) \neq \emptyset$, for some constant $\hat{\epsilon} > 0$. Furthermore, (ii) holds in the stronger form for some $\{f_n\} \in \mathsf{NC}^0$ and $\{g_n\} \notin$ PASSES-SPACE$(n^{\hat{\epsilon}}, \log n)$, where g_n is f_n under an arbitrary permutation of its input and output bits, for some constant $\hat{\epsilon} > 0$.*

Proof. (i) Consider the OR function (i.e. with one output bit) of all bits. Clearly, this can be decided by a finite state automaton. However, the output depends on all input bits and thus it cannot be in NC^0.

(ii) Consider a family of expander graphs. Here is a property of this family.

Claim 5.3.5. *Fix a family of $(n, 3, \epsilon)$-bipartite expanders $\{((L_n, R_n), E)\}_n$. Fix a permutation on L_n such that the permuted L_n is the sequence $L = \langle a_1, a_2, \ldots, a_{|L_n|} \rangle$. For two vertices $u, v \in L$ we say that they are* peers *if there exists $w \in R_n$ connected to both. Then, there exist two sequences A, B subsequences of L, each of length $n^{\epsilon'}$ for some constant $\epsilon' > 0$, such that for every $u \in A$ there exists exactly one peer vertex in B and vice versa (in particular, $|A| = |B|$), and the index (i.e. the subscript wrt the ordering) of every vertex in A is smaller than the index of a vertex in B.*

Proof sketch. Immediate by the expansion property. Consider a (contiguous) subsequence of polynomial size then consider its neighborhood and then the neighborhood of this neighborhood.

Let us give a few more details without getting into detailed calculations. Fix a constant $\alpha < 1$ and consider a contiguous sequence S of length n^α (say in the middle) of L. Let $\Gamma_R := \Gamma(S)$ and $\Gamma_L := \Gamma(\Gamma_R)$. Then, we know that in particular $S \subseteq \Gamma_2$ and furthermore $D := \Gamma_2 - S$ such that $|D| = \Omega(n^\alpha)$ (this is guaranteed by expansion). Now, there are two cases to consider. One where the vertices in D are "balanced" to the left and to the right of S, and the complementary where no significant fraction of D is on both sides. In the first case we consider the elements of D in the left and in the right of S to form the subsequences A and B respectively. In the other case, we construct A with vertices from the leftmost half of S, and we construct B with vertices from the rightmost half of D. □

Define the following function $f : \{0,1\}^n \to \{0,1\}^n$. Given n, consider a graph G from a family of bipartite expander graphs as in Claim 5.3.5. G is the dependency graph of the function f with output bits the XOR of the corresponding input bits.

Suppose that (ii) is false, i.e. $\{g_n\} \in \mathsf{PASSES\text{-}SPACE}(n^{\hat{\epsilon}}, \log n)$ for every $\hat{\epsilon} > 0$. Fix, $\hat{\epsilon} < \epsilon'$, where ϵ' as in Claim 5.3.5. We show that we can decide equality ($\textsc{Equal}_N$) in the 2-party model Communication Complexity model by communicating less than $N-1$ bits, for infinitely many string lengths $N \in \mathbb{Z}^+$.

Let $N = n^{\epsilon'}$. Consider inputs where every bit other than the ones in $A \cup B$ is equal to zero. Here is a protocol for $\textsc{Equal}_N$. Player-1 is given the input corresponding to the bits of A and Player-2 is given the bits corresponding to B. Let M be an $s(n)$-space bounded machine that computes f with $r(n)$ passes. Player-1 carries the simulation up to the point that the input head of M enters B. At that point player-1 sends the $O(\log n)$-bits-long configuration of M to Player-2. Player-2 caries on the simulation of M up to the point where the head enters again A where she sends the configuration of M to Player-1. If M outputs a bit different than 0 then the corresponding player sends a special message (e.g. reserve the first bit as a flag for this in every message the two players exchange) indicating that the two strings are not equal. Hence, in total there are $O(n^{\hat{\epsilon}} \log n)$ bits communicated, which is smaller that $n^{\epsilon'} = N$. □

5.3.3 Impossibility of simulating an $\omega(\log n)$-model by an $O(\log n)$-model

We show that there is no general simulation of a $(\log^k n)$-space transducer by a transducer of smaller memory at an expense a small number of passes over the input. Thus, we cannot base in an obvious way the existence of OWFs on the construction of $\log^{O(1)} n$-streaming OWFs given in Example 5.1.1.

Remark 5.3.6. The application of Communication Complexity we have in mind results in only polylogarithmic (and not polynomial) bounds in the number of passes. One explanation for this is that the ratio of the space of the two models we compare is quite small. We do not

know how to apply a Communication Complexity reduction in this case. It seems that a new Communication Complexity Theory with Auxiliary Inputs should be developed towards this purpose. Technically, we proceed from elementary Information Theory principles by constructing an impossible compress-decompress scheme.

Lemma 5.3.7. *Let $0 \leq \epsilon' < \epsilon$. There exists $\hat{\epsilon} > 0$ and $f \in$ PASSES-SPACE$(1, \log^{1+\epsilon} n)$ such that $f \notin$ PASSES-SPACE$(n^{\hat{\epsilon}}, \log^{1+\epsilon'} n)$.*

Proof. Consider the following family of functions $f_n : \{0,1\}^n \to \{0,1\}^{n/2}$. Let $N = \frac{n}{2\log^{1+\epsilon} n}$ and f defined as follows $f : \alpha_1, \beta_1, \ldots, \alpha_N, \beta_N \mapsto \alpha_1 \oplus \beta_1, \ldots, \alpha_N \oplus \beta_N$, where $|\alpha_i| = |\beta_i| = \log^{1+\epsilon} n$. The TM with memory $O(\log^{1+\epsilon} n)$ in a single pass reads the corresponding α_i and stores it on its worktape and when the head moves to the first bit of β_i it starts outputting the bitwise XOR, $\alpha_i \oplus \beta_i$.

Let M be a $O(\log^{1+\epsilon'} n)$-space bounded TM which computes f_n. Consider the time interval where M starts outputting the first bit of $\alpha_i \oplus \beta_i$ up to the point that outputs the last bit of $\alpha_i \oplus \beta_i$. We show that in this interval M makes at least two passes (it reverses its head at least once). Since this happens for every pair (α_i, β_i) we have an $\Omega(N) = n^{\Omega(1)}$ bound on the number of passes during the computation of M.

Suppose that we choose the input $\alpha_1, \beta_1, \ldots, \alpha_N, \beta_N$ uniformly at random. Let $\mathcal{E}_i$ be the event that the machine makes at least two passes (one reversal) when outputting $\alpha_i \oplus \beta_i$.

Claim 5.3.8. $\Pr[\bar{\mathcal{E}}_i] \leq \frac{1}{N^2}$

This claim implies that $\Pr[\mathcal{E}_1 \wedge \ldots \wedge \mathcal{E}_N] = 1 - \Pr[\bar{\mathcal{E}}_1 \vee \ldots \vee \bar{\mathcal{E}}_N] \geq 1 - \frac{1}{N} > 0$, i.e. there is an input where M makes $\Omega(N)$ passes.

Proof of Claim. Suppose that $\Pr[\bar{\mathcal{E}}_i] > \frac{1}{N^2}$.

Since $\Pr[\bar{\mathcal{E}}_i] > \frac{1}{N^2}$ this means that for at least $2^n/N^2 > 2^{n-2\log n}$ of the input strings, M makes a single pass when outputting $\gamma_i := \alpha_i \oplus \beta_i$. Let this set of strings be S_i. Note that we need to send at least $n - 2\log n$ bits to describe an arbitrary element of S_i. In what follows, we show how to construct an impossible compress-decompress scheme.

Consider two players communicating over a noiseless channel, where Player-I is given a message from S_i and he sends it to Player-II. We show that Player I sends $K' = n - \Omega(\log^{1+\epsilon} n)$ bits to communicate the message (which requires $K = n - 2\log n$ bits), which is a contradiction.

Suppose that Player-I is given $x \in S_i$. He simulates M on input x. Since the input-head of M does not change direction during the outputting of γ_i, wlog assume that it scans the input tape from left to right. We consider two cases: (i) at least half of the bits of γ_i are output before the input-head enters β_i or (ii) otherwise, more than half of the bits are output after the head enters β_i.

Case (i): Player-1 sends to Player-2 a message containing (1) one bit distinguishing between Case (i) and (ii), (2) the configuration of M when the first bit of $\alpha_i \oplus \beta_i$ is outputted, (3) α_i, (4) the second half of β_i, (5) every bit of x except α_i, β_i. In total, he sends

$$1 + O(\log^{1+\epsilon'} n) + \log^{1+\epsilon} n + \frac{\log^{1+\epsilon} n}{2} + n - 2\log^{1+\epsilon} n = n - \Omega(\log^{1+\epsilon} n)$$

many bits to Player-2. Then, Player-2 simulates M and she retrieves the first $\frac{1}{2}\log^{1+\epsilon} n$ bits of β_i, and thus she has a full description of the message with $K' < K$ bits.

Case (ii): Similarly to Case (i) but now instead we transmit without actually sending the second half of α_i. □

□

5.4 Some remarks on the relation of cryptographic hardness and graph-theoretic properties of NC^0 circuits

The work of *Cryptography in* NC^0 brings the combinatorics of computing cryptographic primitives to the level of constant depth and bounded fan-in circuits. It is plausible to assume that a necessary condition for the dependency graph of the circuits is that it is complicated enough in order to preserve cryptographic hardness (security of the primitive). This is implicit in works in the area, but also explicitly stated in [AIK08, Gol00], where "graph-theoretic complexity[8] is associated with expansion". This kind of intuition works against possible constructions of streaming models for Cryptography. Some evidence about this is given in the proof of Lemma 5.3.4. In fact, the simple non-black-box construction is based on the fact that the dependency graph of the cryptographic function is (at a large scale) graph-theoretically very simple. In this section we give some intuition as to why "cryptographic hardness in NC^0" cannot be associated in some obvious way with "graph-theoretic complexity". For the rest of this section "graph-theoretically complex" means high expansion, and graphs are "simplified" by removing edges.

Example 5.4.1. Let f be a OWF computed by NC^0 circuits, and let C be such a circuit with input of length n^ϵ, $\epsilon > 0$. Consider an input $x = x'x''$, where $|x'| = n^\epsilon$ and $|x''| = n - n^\epsilon$, and define $g(x) = g(x'x'') = f(x')x''$. Then, it easily follows that g is also a OWF. If f has good expansion, g does not inherit this property. Note that g has embedded into it a polynomially large expander graph. □

[8] In [AIK08] this is restricted to small $O(\log n)$ "parts" of the circuits.

Here is an interesting question, much weaker than our original question. Suppose that we start with a family of NC^0 circuits that is

1. cryptographically secure and
2. has high expansion

Can we "simplify" circuits in this family by removing wires while maintaining cryptographic hardness? In particular, if we maintain "graph-theoretic complexity" do we also maintain "cryptographic hardness"?

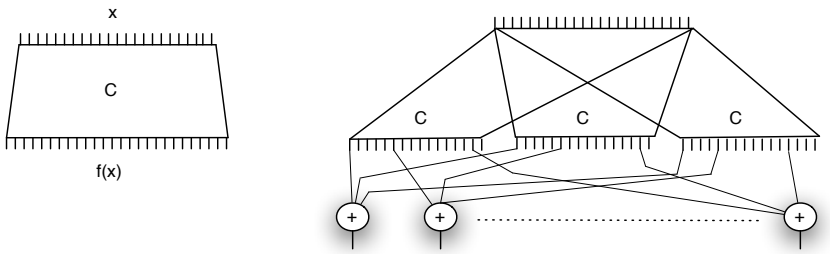
We conclude with two examples indicating a negative answer to the above intuitive question.

Example 5.4.2. Let $f = \{f_n : \{0,1\}^n \to \{0,1\}^n\}$ be a OWF computable by a family $\mathcal{C}'$ of NC^0 circuits. Also, let $\mathcal{C}''$ a family of NC^0 circuits of high expansion, a constant expansion $\epsilon > 0$, that compute the constant $\mathbf{0} = 00\ldots0$ function $g : \{0,1\}^n \to \{0,1\}^n$ (it is immediate that such a family exists using bipartite expanders).

Now, for every input length n construct a circuit C by taking a circuit $C' \in \mathcal{C}$ and a circuit $C'' \in \mathcal{C}''$ with the same input and OR the corresponding outputs. This defines a family of circuits $\mathcal{C}$ which computes f. But, by removing the appropriate wires from a circuit $C \in \mathcal{C}$ we end up with the circuit in $\mathcal{C}''$ which although it has high expansion it computes the zero function. □

Our final example shows that by simplifying a circuit when removing wires it is possible to go from cryptographically hard, to cryptographically insecure, and back to cryptographically hard.

Example 5.4.3. Let C be a circuit from a family of circuits of high expansion that computes a OWF. Construct three copies of the circuit and give them common input. Then, XOR the corresponding output bits; i.e. the i-th output bit of each of the copies. Let this new circuit be C'. Clearly, for every x, $C(x) = C'(x)$. Now, construct C'' by removing all output wires of the first copy to the XOR. For every x, $C''(x) = 0$. Finally, construct C''' by removing from C'' the output wires of the second copy, and thus we are left with an isomorphic copy of C.



□

In-depth investigation of conditions where cryptographic hardness is related to graph-theoretic properties is a very fundamental issue. A first step has been taken in [AIK08].

5.5 Improvements on the logspace streaming computation of the OWF and some conjectures

The naive way of using the NC^0 circuit in order to compute the OWF already achieves poly-logarithmic passes and logarithmic space. Recall, that our unconditional lower bound (Lemma 5.1.4) shows impossibility for a constant number of passes. In fact, depending on the parameter ϵ in the hardness of the OWF we start with, the number of passes can be reduced to $\log^{\epsilon'} n$, for $\epsilon' > 0$ arbitrarily close to 0. To that end we are going to use the Lemma 5.5.2. Note that this lemma is expressed in a slightly more general setting than what we actually need. On the other hand this generality is exactly what we need for stating our most interesting conjectures.

Remark 5.5.1. Lemma 5.3.4 states that we cannot compute an arbitrary NC^0 function in logspace in a streaming manner. This holds even when we allow permutations of its output bits. Let us restrict our attention to functions computable with circuits of small pathwidth. In this case still, in general it is not possible to compute in a streaming manner the same NC^0 function of small pathwidth. This is due to trivial reasons that have to do with the ordering of the inputs and the fact that a transducer outputs bits in order. However, in the sense of Fact 5.2.2 we have that the streaming model is robust for doing Cryptography.

Lemma 5.5.2. *Let $f \in \mathsf{NC}^0$ be computable by the family $\mathcal{F} = \{C_n\}$ of NC^0 circuits whose dependency graphs have given path decompositions of width $w(n)$. Then, for every input length n there exist logspace index-computable permutations π, τ such that $\pi\big(f(\tau(x))\big) \in \mathsf{PASSES\text{-}SPACE}(O(\frac{w(n)}{\log n}), \log n)$.*

Proof. The algorithm that computes $\pi\big(f(\tau(x))\big)$ is a little more complicated than the algorithm that appears in the proof of Lemma 3.5.15 henceforth denoted by $\mathcal{A}$. We proceed by using $\mathcal{A}$. Note that the permutations π and τ are induced by the given path-decomposition.

Since $C \in \mathcal{F}$ is an NC^0 circuit every output gate u has value a boolean function ψ of a constant number of input variables $x_1, \ldots, x_k$, where k depends on the (constant) depth of C. Hence. $\psi(x_1, \ldots, x_k)$ can be expressed as a constant size k-CNF; we abuse notation and we write ψ for this formula.

We perform some syntactic maneuvers before using $\mathcal{A}$. Associate with C a k-CNF formula ϕ which is the conjunction of these constant-size subformulas ψ. We are not interested in the satisfiability of ϕ; we are just interested in the satisfiability of its subformulas ψ (these are the output bits of the function).

Observe that the given path decomposition for the dependency graph of C yields in logspace a decomposition for the incidence graph for ϕ of width $O(w(n))$. Finally, applying the logspace transformation [GP08] from the path decomposition of width $\alpha(n)$ to an ordered formula of diameter $O(\alpha(n))$, we obtain an ordered formula Φ of diameter $O(w(n))$. Two remarks are in order. First Φ admits different models (it has more variables) than ϕ. However, all newly introduced variables appear also as rewritings ($x \leftrightarrow x'$) of the initial variables. Hence, in logspace we can consistently determine the values of the new variables in Φ. The second remark is that the transformation in [GP08] is local in the sense that the satisfiability of the transformed subformulas ψ is maintained.

Now, we are ready to apply $\mathcal{A}$ on Φ. The difference is that the role of the non-deterministic tape is being taken by the input x to the circuit C. In the execution of $\mathcal{A}$ we do not care about the satisfiability of Φ but rather the satisfiability of the (transformed) subformulas ψ. If the subformula is satisfied during the evaluation then we output 1, whereas if it is falsified we output 0. □

Now, we are ready to state our conjecture.

Conjecture 1. *There exists a 2^{n^ϵ}-hard OWF, $\epsilon > 0$ $\iff$ there exists an NC^0 computable OWF function of pathwidth $\log^{\epsilon'} n$, for some $\epsilon' > 0$.*

To the best of our knowledge there is no non-trivial characterization of 2^{n^ϵ}-hard OWFs. None of the directions for this conjecture is trivial. To establish the $\Rightarrow$ direction, by applying [AIK06a, HRV10] it remains to derive an EOWF from a 2^{n^ϵ}-hard OWF. For the other direction we do not have an idea of how to proceed, but we believe that pathwidth (or treewidth) is the most general restriction on the circuits such that the current state of knowledge could establish such a result.

Finally, we also conjecture the following.

Conjecture 2. *There exists an NC^0 computable OWF function of pathwidth $\log^{\epsilon'} n$, for some $\epsilon' > 0$ $\iff$ there exists a logspace streaming computable OWF.*

For this conjecture Lemma 5.5.2 seems to be even more helpful.

All told, we believe that the following can be shown:

there is a 2^{n^ϵ}-hard OWF $\iff$ there is small pathwidth OWF in NC^0 $\iff$ there is logspace streaming computable OWF

5.5.1 Inverting NC^0 functions of small pathwidth

By construction, the OWF in Theorem 5.2.3 can be inverted in time $n^{\log^{O(1)} n}$. In fact, every NC^0 function of such pathwidth can be inverted within this time.

[GP08] gives an algorithm that works in $\mathsf{NSPACE}(w(n))$ for deciding the satisfiability of a k-CNF formula of pathwidth $w(n)$. Since the i-th output gate u_i of an NC^0 circuit is naturally associated with a k-CNF ψ_i of input bits, if the given value of the output is y_i, then the formula $\psi \leftrightarrow y_i$ is satisfied for values of input bits that evaluate to the correct y. Observe that if the circuit has a path decomposition of width $w(n)$ then we can write a k-CNF formula ϕ satisfiable for an $x \in \{0,1\}^n$ such that $f(x) = y$, and ϕ has a logspace computable path decomposition of width $O(w(n))$.

Computing a path-decomposition In practice the path decomposition will be explicit in the description of the circuit. For example, the circuits in the construction which yields Theorem 5.2.3 are described in a way that not only a polylogarithmic bound on the pathwidth but also the diameter of the formula of the inversion algorithm is determined by the layout of the circuit. However, this is not necessary since we can approximate the pathwidth $w(n)$ within a factor $O(\log n)$ and compute a path decomposition, by an approximation algorithm that runs in time $n^{O(1)}2^{O(w(n))}$. For example, one can modify the constant approximation algorithm of Robertson and Seymour [RS95] à la Alekhnovitch and Razborov [AR02].

5.6 Discussion

It is conceivable that Cryptography in Streaming Models finds application in practical streaming or on-line settings. In fact, although it relies on Cryptography in NC^0 the streaming setting seems to be more applicable than the NC^0 one - hence in some sense we answer an intuitive question stated in the works for Cryptography in NC^0, regarding practicality. An interesting future problem is to get rid of the EOWF assumption. If we can base the existence of an EOWF on very hard OWFs then on one hand this strengthens our results, but also implies the possibility of Cryptography in NC^0 based on 2^{n^ϵ}-hard OWFs.

In this chapter we constructed OWFs. The question of $O(\log n)$-streaming computation of more involved primitives, and in particular PRGs, is open. From a practical point of view it is worth investigating constructions based on concrete intractability assumptions, without making further assumptions about their hardness. For example, we would like to construct in a streaming way PRGs of polynomial stretch. To that end perhaps we can base streaming computation of linear-stretch PRGs on an assumption by Alekhnovitch [Ale03] about the indistiguishability of two distributions of matrices (a similar, equivalent assumption is used in [AIK08]). Observe that if a PRG has stretch ϵn and it is computable in logspace, then by a sequential composition we can obtain any polynomial stretch with no expense on the number of passes over the seed.

Our current main motivation is the theoretical investigation of such models. A concrete goal

is to understand better the relation with Cryptography in NC^0. There are several open questions which when realized in the context of streaming models or in Cryptography in NC^0 seem more feasible. For example, can we make progress in giving a precise answer in hardness amplification through direct product constructions when the functions are computable in a streaming model? Also, can we say something more precise, and conceptually more interesting (than in the general polytime case) about the existence of a universal OWF ("one-wayest function") computed by streaming models? Finally, our more ambitious goal is to characterize 2^{n^ϵ}-hard OWFs through streaming models. This would be an interesting characterization, i.e. a characterization not mentioning "2^{n^ϵ}".

Bibliography

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[AIK06a] B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006. (also CCC'05).

[AIK06b] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput*, 36(4):845–888, 2006. (also FOCS'04).

[AIK07] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography with constant input locality. In *CRYPTO'07*, number 4622 in Lecture Notes in Computer Science, pages 92–110. Springer, 2007.

[AIK08] B. Applebaum, Y. Ishai, and E. Kushilevitz. On pseudorandom generators with linear stretch in NC^0. *Comput. Complexity*, 17(1):38–69, 2008. (also RANDOM'06).

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004.

[Ale03] M. Alekhnovich. More on average case vs approximation complexity. In IEEE, editor, *44th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2003, 11–14 October 2003, Cambridge, Massachusetts*, pages 298–307, pub-IEEE:adr, 2003. IEEE Computer Society Press.

[All85] E. Allender. *Invertible Functions*. PhD thesis, Georgia Institute of Technology, 1985.

[All89] E. W. Allender. P-uniform circuit complexity. *J. Assoc. Comput. Mach.*, 36(4):912–928, 1989.

[AMS96] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *ACM Symposium on Theory of Computing (STOC)*. ACM, 1996.

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[AR02] M. Alekhnovich and A. A. Razborov. Satisfiability, branch-width and tseitin tautologies. In *Foundations of Computer Science (FOCS)*. IEEE, 2002.

[BCD+89] A. Borodin, S. A. Cook, P. Dymond, L. Ruzzo, and M. Tompa. Two applications of inductive counting for complementation problems. *SICOMP: SIAM Journal on Computing*, 18, 1989.

[BCHK06] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. of Computing (SICOMP)*, 36(5):915–942, 2006.

[BCOP04] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Eurocrypt '04*, 2004.

[BDS+03] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H.C. Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196, 2003.

[BF01] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Crypto 2001*, volume 2139 of *LNCS*, pages 213–29. Springer-Verlag, 2001.

[BF03] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. extended abstract in Crypto 2001.

[BGH82] A . Borodin, J. von zur Gathen, and J. E. Hopcroft. Fast parallel matrix and GCD computations. *Information and Control*, 52(3):241–256, 1982. (also FOCS'82).

[BGH07] D. Boneh, C. Gentry, and M. Hamburg. Space-efficient identity based encryption without pairings. In *FOCS'2007*, 2007.

[BGW05] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO '05*, pages 258–275, 2005.

[BHN08] P. Beame and D.T. Huynh-Ngoc. On the value of multiple read/write streams for approximating frequency moments. In *FOCS 2008*, 2008.

[BJR07] P. Beame, T. S. Jayram, and A. Rudra. Lower bounds for randomized read/write stream algorithms. In *STOC 2007*, 2007.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequence of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984. (also FOCS'82).

[BMG07] B. Barak and M. Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *FOCS*, pages 680–688. IEEE Computer Society, 2007.

[BNS92] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992.

[Bon98] D. Boneh. The Decision Diffie-Hellman problem. In *Proc. Third Algorithmic Number Theory Symp*, volume LNCS Vol. 1423, pages 48–63. Springer-Verlag, 1998.

[BPR+08] D. Boneh, P. A. Papakonstantinou, C. W. Rackoff, Y. Vahlis, and B. Waters. On the impossibility of identity based encryption from trapdoor permutations. In *Foundations of Computer Science (FOCS)*. IEEE, 2008.

[BQ09] A. Bogdanov and Y. Qiao. On the security of Goldreich's one-way function. In *APPROX-RANDOM*, pages 392–405, 2009.

[CHK03] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Eurocrypt 2003*, volume 2656 of *LNCS*. Springer, 2003.

[CKS03] A. Chakrabarti, S. Khot, and X. Sun. Near-optimal lower bounds on the multi-party communication complexity of set disjointness. In *IEEE Conference on Computational Complexity (CCC)*, pages 107–117. IEEE Computer Society, 2003.

[CM01] M. Cryan and P. Bro Miltersen. On pseudorandom generators in NC. In Jiri Sgall, Ales Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science 2001 (MFCS'01)*, volume 2136 of *Lecture Notes in Computer Science*, pages 272–284. Springer, 2001.

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *The 8th IMA International Conference on Cryptography and Coding*, pages 26–8, 2001.

[Coo71] S. A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *J. Assoc. Comput. Mach.*, 18:4–18, 1971.

[CS98] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO*. Springer, 1998.

[DF02] Y. Dodis and N. Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[DK00] Ding-Zhu Du and Ker-I Ko. *Theory of Computational Complexity*. John Wiley and Sons, 2000.

[DP10] M. David and P. A. Papakonstantinou. Trade-off lower bounds for stack machines. In *IEEE Conference on Computational Complexity (CCC)*, 2010. (to appear).

[DPS09a] M. David, P. A. Papakonstantinou, and A. Sidiropoulos. How strong is Nisan's pseudorandom generator? manuscript, 2009.

[DPS09b] M. David, P. A. Papakonstantinou, and A. Sidiropoulos. Polynomial time with restricted use of randomness. Technical Report 09-039, Electronic Colloquium on Computational Complexity (ECCC), April 2009.

[ElG85] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31:469–472, 1985.

[ERIS91] J. Edmonds, S. Rudich, R. Impagliazzo, and J. Sgall. Communication complexity towards lower bounds on circuit depth. In *FOCS*, pages 249–257, 1991.

[ES35] P. Erdős and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

[FG06] J. Flum and M. Grohe. *Parameterized complexity theory*. Springer-Verlag, Berlin, 2006.

[FK97] U. Feige and J. Kilian. On limited versus polynomial nondeterminism. *Chicago J. Theor. Comp. Sci.*, 1997.

[GGKT05] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM J. Comput*, 35(1):217–246, 2005.

[GHS06] M. Grohe, A. Hernich, and N. Schweikardt. Randomized computations on large data sets: Tight lower bounds. In *PODC06*, 2006.

[GKM+00] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS*, pages 325–335, 2000.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *ACM symposium on Theory of computing – STOC'89*, pages 25–32, Berlin, 1989. ACM.

[GMM07] Y. Gertner, T. Malkin, and S. Myers. Towards a separation of semantic and CCA security for public key encryption. In *TCC 2007*, 2007.

[Gol00] O. Goldreich. Candidate one-way functions based on expander graphs. Technical report, ECCC, 2000.

[Gol01] O. Goldreich. *Foundations of Cryptography (Volume 1)*. Cambridge University Press, 2001.

[GP08] K. Georgiou and P. A. Papakonstantinou. Complexity and algorithms for well-structured k-CNF formulas. In *Theory and Applications of Satisfiability Testing - SAT 2008*, 2008.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC '08*, 2008.

[Gro03] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. *J. ACM*, 54(1):(electronic), 2007 (also FOCS'03).

[GS05] M. Grohe and N. Schweikardt. Lower bounds for sorting with few random accesses to external memory. In *PODS 2005*, 2005.

[HG91] J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1:113–129, 1991.

[HHRS07] I. Haitner, J. J. Hoch, O. Reingold, and G. Segev. Finding collisions in interactive protocols - A tight lower bound on the round complexity of statistically-hiding commitments. In *FOCS*, pages 669–679. IEEE Computer Society, 2007.

[HILL99] J. Hastad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HLW06] S. Hoory, N. Linial, and A. Wgderson. Expander graphs and their applications. *BAMS: Bulletin of the American Mathematical Society*, 43:439–561, 2006.

[HRV10] I. Haitner, O. Reingold, and S. Vadhan. Efficiency improvements in constructions of pseudorandom generators from one-way functions. In *STOC'10*, 2010. (to appear).

[HS08] A. Hernich and N. Schweikardt. Reversal complexity revisited. *Theoret. Comput. Sci.*, 401(1-3):191–205, 2008.

[IK00] Y. Ishai and E. Kushilevitz. Randomizing polynomials: a new representation with applications to round-efficient secure computation. In Danielle C. Young, editor, *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 294–304, Los Alamitos, California, November 12–14 2000. IEEE Computer Society.

[IN89] R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. In *30th Annual Symposium on Foundations of Computer Science (FOCS '89)*, pages 236–241, Washington - Brussels - Tokyo, October 1989. IEEE.

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *ACM Symposium on Theory of Computing (STOC)*. ACM Press, 1994.

[IPZ98] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comp. Sys. Sci.*, 63(4):512–530, 2001 (also FOCS'98).

[IR88] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Advances in Cryptology—CRYPTO*, pages 8–26. Springer, 1988.

[IW97] R. Impagliazzo and A. Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC 1997*, 1997.

[JP10] A. Juma and P. A. Papakonstantinou. Streaming and non-adaptive cryptography. manuscript, 2010.

[KGY89] M. Kharitonov, A. V. Goldberg, and M. Yung. Lower bounds for pseudorandom number generators. In *FOCS*, pages 242–247, 1989.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complexity*, 13(1-2):1–46, 2004. (also STOC'03).

[KM07] N. Koblitz and A. Menezes. Another look at generic groups. *Advances in Mathematics of Communications*, 1:13–28, 2007.

[KN97] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, New York, NY, USA, 1997.

[KRW95] M. Karchmer, R. Raz, and A. Wigderson. Super-logarithmic depth lower bounds via direct sum in communication complexity. *Computational Complexity*, 5:191–204, 1995.

[KST99] J. H. Kim, D. Simon, and P. Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *FOCS '99*, 1999.

[KV85] M. Karpinski and R. Verbeek. There is no polynomial deterministic space simulation of probabilistic space with a two-way random-tape generator. *Inform. and Control*, 67(1-3):158–162, 1985.

[KW90] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. *SIAM Journal on Computing*, 3:255–265, 2 1990.

[Lys02] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *Crypto '02*, 2002.

[Mar07] D. Marx. Can you beat treewidth? In *FOCS'07*, pages 169–179, 2007.

[MMN10] F. Magniez, C. Mathieu, and A. Nayak. Recognizing well-parenthesized expressions in the streaming model. In *STOC'10*, 2010. (to appear).

[MP80] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theor. Comput. Sci.*, 12:315–323, 1980. (also FOCS'79).

[Mul87] K. Mulmuley. A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. *Combinatorica*, 7(1):101–104, 1987. (also STOC'86).

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[Nis93a] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoret. Comput. Sci.*, 107(1):135–144, 1993. Structure in complexity theory (Barcelona, 1990).

[Nis93b] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoret. Comput. Sci.*, 107(1):135–144, 1993. Structure in complexity theory (Barcelona, 1990).

[NW88] N. Nisan and A. Wigderson. Hardness vs. randomness. *J. Comput. System Sci.*, 49(2):149–167, 1994 (preliminary version in FOCS 1988).

[Pap09] P. A. Papakonstantinou. A note on width-parameterized SAT: an exact machine model characterization. *Information Processing Letters*, 110(1):8–12, 2009.

[Pap10] P. A. Papakonstantinou. Cryptography in NC^0, PolyLogSpace vs BPP, and limited verification of NP witnesses. (manuscript) http://www.cs.toronto.edu/∼papakons/thesis_add_on.pdf, 2010.

[PRVW10] P. A. Papakonstantinou, C. W. Rackoff, Y. Vahlis, and B. Waters. On the limits of the Decisional Diffie Hellman assumption. (work in progress), 2010.

[RM97] R. Raz and P. McKenzie. Separation of the monotone nc hierarchy. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 234, Washington, DC, USA, 1997. IEEE Computer Society.

[RS95] N. Robertson and P. D. Seymour. Graph minors: XIII. the disjoint paths problem. *J. Comb. Theory Series B*, 63(1):65–110, 1995.

[RTV04] O. Reingold, L. Trevisan, and S. Vadhan. Notions of reducibility between cryptographic primitives. In *TCC '04*, LNCS, pages 1–20, 2004.

[Ruz80] W. L. Ruzzo. Tree-size bounded alternation. *J. Comput. System Sci.*, 21(2):218–235, 1980.

[Ruz81] W. L. Ruzzo. On uniform circuit complexity. *J. Comput. System Sci.*, 22(3):365–383, 1981. Special issue dedicated to Michael Machtey.

[RW92] R. Raz and A. Wigderson. Monotone circuits for matching require linear depth. *Journal of the ACM*, 39:736–744, 1992.

[Sha85] A. Shamir. Identity-based cryptosystem and signature scheme. In *Advances in Cryptology – CRYPTO ' 84*. Springer, 1985.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.

[Sim98] D. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions. In *Eurocrypt '98*, 1998.

[STV99] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. System Sci.*, 62(2):236–266, 2001 (preliminary version in STOC 1999). Special issue on the Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999).

[Sze03] S. Szeider. On fixed-parameter tractable parameterizations of SAT. In *SAT'03*, pages 188–202, 2003.

[Uma03] C. Umans. Pseudo-random generators for all hardnesses. *J. Comput. System Sci.*, 67(2):419–440, 2003. Special issue on STOC2002 (Montreal, QC).

[Ven91] H. Venkateswaran. Properties that characterize LOGCFL. *J. Comput. System Sci.*, 43(2):380–404, 1991.

[Ven06] H. Venkateswaran. Derandomization of probabilistic auxiliary pushdown automata classes. In *IEEE Conference on Computational Complexity (CCC)*, volume 21, 2006.

[Vol98] H. Vollmer. *Introduction to Circuit Complexity.* Springer, 1998.

[Yao79] A. C. Yao. Some complexity questions related to distributive computing. In *ACM Symposium on Theory of Computing (STOC).* ACM, 1979.

[Yao82] A. C. Yao. Theory and applications of trapdoor functions. In *Foundations of Computer Science (FOCS).* IEEE, 1982.

[YFDL04] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In Birgit Pfitzmann, editor, *ACM Conference on Computer and Communications Security 2004*, pages 354–63, 2004.

[YY94] X. Yu and M. Yung. Space lower-bounds for pseudorandom-generators. In *Proceedings of the 9th Annual Conference on Structure in Complexity Theory (predecessor of CCC)*, pages 186–197, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.