UNIVERSITY OF CALIFORNIA, SAN DIEGO

New Proofs of (New) Direct Product Theorems

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in
Computer Science and Engineering

by

Ragesh Jaiswal

Committee in charge:

Professor Russell Impagliazzo, Chair
Professor Sam Buss
Professor Fan Chung Graham
Professor Nolan Wallach
Professor Sanjoy Dasgupta

2008

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I am greatly indebted to my advisor, Russell Impagliazzo, for his patient and careful guidance. His immense clarity, deep intuition, breadth of knowledge and creative thinking has influenced me a lot.

I am indebted to Russell Impagliazzo, Valentine Kabanets and Avi Wigderson for their collaboration, and insights which has helped to shape my research work.

I am thankful to the Theory group at UC San Diego. I have learned a lot from the conversations with my fellow graduate students, including Kirill Levchenko, Yi-Kai Liu, Vadim Lyubashevsky, Chris Calabro, and Sashka Davis. I am also thankful to the Indian group at UCSD for making my stay a fun experience.

Last but not the least, I am indebted to my family for their unquestioning and unwavering support for all my pursuits.

Chapter II is, in part, based on the paper "Approximately List-decoding Direct Product Codes and Uniform Hardness Amplication" co-authored with Russell Impagliazzo and Valentine Kabanets. In Proceedings of the Forty-Seventh Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pages 187–196, 2006.

A summary of the results in Chapter III appeared in the paper "Uniform Direct Product Theorems: Simplified, Optimized, and Derandomized" co-authored with Russell Impagliazzo, Valentine Kabanets and Avi Wigderson. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC'08), pages 579–588, 2008.

Chapter V is, in part, based on the paper "Chernoff-type Direct Product Theorems" co-authored with Russell Impagliazzo and Valentine Kabanets. In Advances in Cryptology - CRYPTO 2007, Twenty-Seventh Annual International Cryptology Conference, pages 500–516, 2007.

VITA

1980	Born, Jamshedpur, India
2003	B.Tech., Indian Institute of Technology Kanpur, India
2005	M.S., University of California San Diego, USA
2003–2008	Research Assistant, Department of Computer Science and Engineering, University of California San Diego, USA
2008	PhD., University of California San Diego, USA

PUBLICATIONS

Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. "Approximately List-decoding Direct Product Codes and Uniform Hardness Amplication". In Proceedings of the Forty-Seventh Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pages 187–196, 2006.

Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. "Chernoff-type direct product theorems". In Advances in Cryptology - CRYPTO 2007, Twenty-Seventh Annual International Cryptology Conference, pages 500–516, 2007.

Russell Impagliazzol, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. "Uniform direct-product theorems: Simplied, optimized, and derandomized". In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC'08), pages 579–588, 2008.

FIELDS OF STUDY

Major Field: Computer Science
Studies in Theoretical Computer Science.
Professors Russell Impagliazzo

ABSTRACT OF THE DISSERTATION

New Proofs of (New) Direct Product Theorems

by

Ragesh Jaiswal
Doctor of Philosophy in Computer Science and Engineering
University of California, San Diego, 2008
Professor Russell Impagliazzo, Chair

In this Dissertation we give an alternate proof of the well known Direct Product Theorem which in contrast to the previous proofs, achieves optimal values for the related parameters. We also obtain a stronger version of the Direct Product Theorem which is motivated by some interesting applications in Cryptography.

Direct Product Theorems are formal statements of the intuition: "if solving one instance of a problem is hard, then solving multiple instances is even harder". For example, a Direct Product Theorem with respect to bounded size circuits computing a function is a statement of the form: "if a function f is hard to compute on average for small size circuits, then $f^k(x_1, ..., x_k) \overset{def}{=} f(x_1), ..., f(x_k)$ is even harder on average for certain smaller size circuits". The proof of the such a statement is by contradiction, that is, we start with a circuit which computes f^k on some non-negligible fraction of the inputs and then use this circuit to construct another circuit which computes f on almost all inputs. By viewing such a constructive proof as decoding certain error-correcting code, it was independently observed by Trevisan [Tre03] and Impagliazzo [Imp02] that constructing a single circuit is not possible in general. Instead, we can only hope to construct a list of circuits such that one of them computes f on almost all inputs. This makes the list size an important parameter of the Theorem which can be minimized. In a sequence of results [IJK06] and [IJKW08], we achieve optimal value of the list size which

is a substantial improvement compared to previous proofs of the Theorem. In particular, this new version can be applied to uniform models of computation (e.g. randomized algorithms) whereas all previous versions applied only to nonuniform models (e.g. circuits). This proof is presented in Chapter III.

Consider the following stronger and a more general version of the previous Direct Product Theorem statement: "if a problem is hard to solve on average, then solving more than the expected fraction of problem instances from a pool of multiple independently chosen instances becomes even harder". Such statements are useful in cryptographic settings where the goal is to amplify the gap between the ability of legitimate users to solve a type of problem and that of attackers. We call such statements "Chernoff-type Direct Product Theorems" and prove such a statement for a very general setting in [IJK07]. This is presented in Chapter V.

I

Introduction

Direct product theorems are formal statements of the intuition:
If a problem is hard to solve on the average, then solving multiple instances of the problem is even harder.

In order to make the above informal statement more precise, we will need to define what the problem is, the entity solving the problem, and what do we mean by hard on average? The problem could be as simply defined as computing some function or could involve a more complex setting where the goal is to generate an attack for certain cryptographic protocol. The entity solving the problem could be a bounded size circuit or an algorithm with bounded running time. Finally, a problem is hard on average if a bounded resource solver fails to solve the problem on some noticeably large number of instances of the problem. For the initial part of this Dissertation, we will consider the problem of computing functions by bounded size circuits and algorithms with bounded running time.

Before we start analyzing more formal statements of the direct product theorem, we begin by looking at some of the places where such statements are useful. First, we take a look at the general organization of this Dissertation.

Organization of this Dissertation In the remainder of this chapter, we will discuss application areas of direct product theorems and some proof techniques.

In Chapter II, we look at a coding theoretic view of the direct product theorem and establish bounds for the parameters involved in the theorem. In Chapter III we give our new proof of the theorem which achieves optimal bounds which were established in Chapter II. In Chapter IV, we discuss an applications of such uniform version of direct product theorems. Chapter V discusses a stronger and more general version of the theorem which we call "Chernoff-type Direct Product Theorem". Finally, in Chapter VI we discuss some interesting applications of direct product theorems.

I.A Applications of Direct Product Theorems

I.A.1 Average-case Complexity

In the 1970s, a significant amount of progress was made in showing that a large number of naturally occurring problems belong to a special class called NP. For example, a very important problem within NP is *satisfiability.* Here an instance of the problem is a formula $\psi(x_1, ..., x_n)$ over boolean variables $x_1, ..., x_n$ and the goal is to tell whether there is an assignment to these variable which makes ψ true. Some problems in the class NP are believed to be hard for polynomial time algorithms in the *worst case.* This essentially means that no algorithm would be able to solve a problem in the class NP on *all* instances, where the algorithm is only allowed time which is some polynomial in the size of the instance. This is basically the P versus NP question which is one of the most important open problems in Theoretical Computer Science.

Even if a problem is hard in the worst case, it might be easy to solve on "typical" instances, or in other words on instances that occur naturally for that problem. This suggests that a better notion for studying hardness of problems would be to evaluate the behavior of efficient algorithms against problem instances which are themselves generated efficiently. This leads to a structural theory of Average-case complexity of problems, the foundation of which was laid by Levin

[Lev86]. Much work was been done building on this foundation. One important question that arises in Average-case complexity is that of *Hardness Amplification.* Here we are interested in how the degrees of average-case hardness of problems within a certain complexity class, say NP, are related. More specifically, say we have a problem within NP which is "mildly" hard on the average for efficient algorithms, then does this imply that there is another problem within NP which is very hard on average for efficient algorithms? Note the similarity of the previous statement with the direct product theorem statement. Indeed, direct product theorems play a very crucial role in addressing such questions. We will see this formal connection when we study Uniform Hardness Amplification in Chapter IV of this Dissertation.

We encourage the readers to look at [Imp95b, BT06] for some some excellent survey on average-case complexity.

I.A.2 Cryptography

Cryptography is built upon problems which are believed to be hard on the average, or in other words, are intractable on most instances. More specifically, most of the Cryptographic Primitives [1] are based on the assumption that *one-way functions* exists. These are functions which are hard to invert on the average. This essentially means that for most inputs x, given $f(x)$, it is hard to compute any element of $f^{-1}(f(x))$. So, the parameter of interest is the fraction of inputs for which a function cannot be inverted by efficient algorithms. Given this, a natural question to ask is whether there is a generic method to amplify this parameter. That is, if there is a one-way function which is "weak" in the sense that it is hard to invert the function for some non-negligible number of inputs, then does that imply that there is a "strong" one-way function (hard to invert on almost all inputs)? Moreover, is there a generic way to construct a strong one-way function from weak ones? Once again note the similarity of the previous statement to the

[1] these are very basic components used to accomplish various Cryptographic tasks.

statement of the direct product theorem. Indeed, weak one-way functions implies strong ones [Gol01, Yao82] and the proof arguments essentially involves proving a direct product theorem.

Direct product theorems have more direct applications in Cryptography. Consider the following examples of cryptographic challenges:

1. *Two-round protocol involving puzzles*: Consider a simple two-round, two-party protocol involving a prover and a verifier. The verifier sends a randomly generated puzzle as a challenge to the verifier. The verifier needs to solve this puzzle in order to get accepted. An example is the CPATCHA protocol where the prover is trying to convince the verifier that it is a human user by answering randomly generated CAPTCHA puzzles. These puzzles are usually based on some hard artificial intelligence problem like determining a distorted text.

2. *Digital Signatures (DS) and Message Authentication Codes (MAC)*: The challenge for the adversary is to generate a DS/MAC for any message after obtaining DSs/MACs for a few messages. The adversary is said to succeed if it is able to generate a correct DS/MAC for many secret keys.

The harder it is for an adversary to meet these challenges the stronger the cryptographic protocol is. So an important question for such cryptographic challenges is whether we can amplify the difficulty of an adversary by asking it to solve multiple challenges in parallel instead of a single challenge. In many cases, it can be shown that parallel repetition of a cryptographic protocol indeed achieves this property. The main work involved is formulating and proving a direct product theorem. We will see such results in Chapter V of this Dissertation.

I.A.3 Derandomization

Randomness is a very useful paradigm in algorithm design. In the past, it has been used to construct efficient algorithms for problems for which no efficient deterministic (not using any randomness) algorithm was known. On the other hand, there have been a number of instances where such problems have later been *derandomized*. This essentially means that an efficient deterministic algorithm was found for the problem. One classical example is the derandomization of the Primality Test[2] due to Agrawal, Kayal, and Saxena [AKS04]. This raises a fundamental question "is randomness useful as far as computational efficiency is concerened?". One way to address this question, when aiming for a negative answer, is to try to construct a universal procedure that can be used to derandomize *any* probabilistic computation.

A randomized algorithm is different from a deterministic algorithm in that it takes as input a random string in addition to the input. A natural way to derandomize any probabilistic computation is to try out all random strings, but then it no longer remains efficient. This is because the size of the random input is allowed to be proportional to the size of the input. *Pseudorandom generators* which have been studied in Cryptographic settings are known to stretch a small truly random string, called *seed*, to a longer random "looking" string. This essentially means that no efficient algorithm would be able to tell apart a truly random string from the output of the pseudorandom generator. Given an appropriate pseudorandom generator we can give the output of this pseudorandom generator (instead of truly random string) to the probabilistic algorithm and then try out all possible random seeds. The hope is that the number of distinct random seeds are small enough to keep the computation efficient. Unfortunately, as in Cryptography where the existence of a pseudorandom generator is based on the existence of one-way functions, the pseudorandom generator required in this setting is also conditional. The existence of a generator required for derandomization is based on the average-case hardness of the complexity class EXP (problems which can be solved in deterministic time $2^{poly(n)}$). [NW94] used the above sequence of arguments to show, in some sense, that average-case hardness can be traded for randomness.

[2]here the problem is to check whether a given number of prime.

There was further interest in obtaining a similar tradeoff starting from a worst-case assumption. This was achieved by [BFNW93] which showed a worst-case to average-case reduction for the class EXP. This essentially means that an instance of any problem in EXP can be solved by solving randomly chosen instances. Note that this also means that if a problem in EXP is hard in the worst-case then it is also hard in the average-case. The worst-case to average-case reduction in [BFNW93], however, gave a function which was only mildly hard on the average. XOR Lemma, which is closely related to the direct product theorem, was used to amplify the average-case hardness but the parameters fell short of the requirements of the [NW94] construction and only partial derandomization was obtained. Finally, [IW97] overcame this by giving a derandomized version of the XOR Lemma. In Section I.B.1 we will show how the XOR Lemma, which plays an important part in the above derandomization results, is related to the direct product theorem.

A lot of progress in derandomization has been made more recently. Interested readers are encouraged to look at [Kab02] and [Imp02] for nice surveys on this topic.

I.A.4 Error-correcting Codes

An Error-correcting code is a mapping $C : \Sigma^m \rightarrow \Sigma^n$ (Σ is a finite set called the alphabet for the code) such that given a string which is sufficiently close to the encoding of a message, it is possible to output this message. The minimum distance (number of places where two strings disagree) between the encoding of two messages is called the distance of the code C. A necessary and sufficient condition for the above decoding property to hold is that this distance is not too small. In addition to the distance being large, we would ideally like to keep the length of the codeword small. So, the rate of the code which is defined to be the ratio m/n is another important parameter of the encoding which we would like to minimize. Efficiency of decoding is also an important issue that should be addressed. Error-correcting codes have numerous applications ranging from practical areas like Communications and Data Storage to theoretical areas like Average-case complexity and Derandomization.

One simple but very useful coding construction is the Hadamard code. Here the alphabet is binary, that is, $\Sigma = \{0, 1\}$. Given a message $x \in \{0, 1\}^m$ of size m its codeword $C(x)$ is of size $n = 2^m$. Suppose we index the codeword by a string $r \in \{0, 1\}^m$, then the bit of codeword indexed by r is given by $C(x)[r] = \langle x, r\rangle$, where $\langle ., .\rangle$ denotes the inner product of two binary strings. Note that this code has a very large distance $n/2$ but an extremely poor rate $m/2^m$. Let us try to improve the rate by changing the code slightly. Let the codeword of size $\binom{m}{k}$, be indexed by string $r \in \{0, 1\}^m$ such that hamming weight[3] of r is exactly k. Then the bit of the codeword indexed by a string $r \in \{0, 1\}^m$ of hamming weight k is given by $C(x)[r] = \langle x, r\rangle$. This is known as the k-truncated hadamard code. Note that this code has poor distance property since any two messages that are close to each other will have almost the same encodings. On the other hand the rate of the code is large compared to the hadamard code.

From the point of view of this Dissertation, this code is interesting because of its close connection with the direct product theorem. Given a message $x \in \{0, 1\}^m$ we can interpret this message as the truth table of a function $f : [m] \rightarrow \{0, 1\}$ such that $\forall i \in [1...m], f(i) = x[i]$. In that case the bit of the codeword indexed by $r \in \{0, 1\}^m$ with hamming weight k is nothing but $C(x)[r] = f^{\oplus k}(i_1, ..., i_k) \overset{def}{=} f(i_1) \oplus f(i_1) \ldots \oplus f(i_k)$, where $i_1, ..., i_k$ are the k indices where r has a non-zero value. If we can show that average-case hardness of f implies stronger average-case hardness of $f^{\oplus k}$ then this could give us a decoding algorithm for the k-truncated hadamard code. This is because the average-case hardness amplification proof is usually by contradiction. That is, we start with the assumption that we know the correct value of $f^{\oplus k}$ on some non-negligible fraction of inputs and then try to compute f on most inputs. So, basically given a corrupted codeword we have a procedure that reconstructs the message approximately. As

[3]number of 1's in the string

we will see in the next section, the proof of the above hardness amplification goes through a direct product theorem.

We will discuss these codes and their connection with the direct product theorem in detail in the next Chapter.

I.B A Direct Product Theorem

In order to initiate the discussion and begin formal analysis, we look at a direct product theorem in this section. Even though this is the simplest of the direct product theorems that we will study in this Dissertation, the analysis will bring out most of the underlying concepts, techniques, and issues. We consider hardness of boolean functions against boolean circuits. We will need the following definitions.

Definition I.B.1 (Circuit)**.** A Circuit C with n inputs and m outputs is a directed acyclic graph. It has n inputs nodes with no incoming edges and m output nodes with no outgoing edges. All the other nodes are called boolean gates and are labeled $\vee$, $\wedge$, and $\neg$ (denoting a boolean OR, AND, and negation). The nodes labeled $\vee$ and $\wedge$ have two incoming nodes and nodes labeled $\neg$ have only one incoming edge. The size of the circuit, which is denoted by $|C|$, is total number of nodes in this directed acyclic graph.

A *Boolean Circuit* is a circuit with a single output.

The boolean circuit defined above implements a boolean function. For a boolean function $f : \{0,1\}^n \to \{0,1\}$, we say that a circuit C computes this function if $\forall x \in \{0,1\}^n, C(x) = f(x)$. Note that any such function f can be computed by a large circuit of size $O(n2^n)$ [4] . A natural question to ask is whether any such function can also be computed by circuits of smaller size, say circuits of size $poly(n)$ for some polynomial poly. The answer turns out to be negative by the

[4]This can easily be seen by expressing f as a boolean formula in conjunctive normal form and then constructing the circuit which implements this boolean formula.

following simple counting argument due to [RS42]: there could be at most $2^{3S\log S}$ different circuits[5] with S gates, whereas there are 2^{2^n} different boolean functions. This means that there are lots of boolean functions which are not computable by any boolean circuit with size as large as $2^n/(30n)$. Which means that there are functions which cannot be computed by a boolean circuits of any (fixed) polynomial size. Note that this simple argument gives us our first unconditional hardness result. This however is a worst-case hardness result for boolean functions against bounded size boolean circuits. The next natural question to ask is whether we can get a stronger, average-case hardness result. For instance consider the following stronger statement: is there a boolean function such that no polynomial size circuit is able to compute the function on most inputs. In other words, any polynomial size circuit C disagrees with this function on non-negligible, say constant fraction of inputs. We again use counting arguments to show that there are functions which are hard on the average for polynomial size circuits. Let us first formally define average-case hardness of boolean functions.

Definition I.B.2 (Average-case hardness of boolean functions)**.** Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a boolean function, $0 < \delta \leq 1$, and $s \in \mathbb{N}$. f is said to be δ-hard for circuits of size s if for any circuit C of size at most s we have

$$Pr_{x\in\{0,1\}^n}[f(x) = C(x)] < (1-\delta)$$

We will need the following definitions and lemma to show an unconditional average-case hardness result.

Definition I.B.3 (Hamming Distance)**.** Let $N \in \mathbb{N}$. The hamming distance between any two string $x, y \in \{0,1\}^N$ which is denoted by $\Delta(x,y)$ is defined as

$$\Delta(x,y) = |\{i : x[i] \neq y[i], i \in \{1...N\}\}|$$

Definition I.B.4 (Volume of a Hamming Ball)**.** Let $N \in \mathbb{N}$. Given a binary string $c \in \{0,1\}^N$, a hamming ball of radius $r \in \{1, ..., N\}$ centered at s is defined to be

[5] each gate accepts at most 2 inputs and 1 output and there are S gates which means there could be at most $(S^3)^S$ possible acyclic graphs denoting the circuit

the subset of string $S \subseteq \{0,1\}^N$ such that $\forall x \in S, \Delta(x,c) \leq r$. The Volume of such a hamming ball, denoted by $V(c,r)$ is the size of the set S.

Next, we state the following well known Lemma without proof.

Lemma I.B.5 (Folklore)**.** *Let $N \in \mathbb{N}$ and $0 < \delta \leq 1/2$ is some constant. For any $c \in \{0,1\}^N$, $V(c, \delta N) \leq 2^{H(\delta)N}$, where $H(\delta) = (1-\delta)\log 1/(1-\delta) + \delta \log 1/\delta$.*

The following Theorem uses counting argument to show that there exists functions which are hard on average for polynomial size circuits.

Theorem I.B.6. *Let $n \in \mathbb{N}$. There exists a boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that f is $(1/4)$-hard for circuits of size $2^n/(30n)$.*

Proof. Let $N = 2^n$. Let F be the set of functions mapping n bits to a single bit. Then $|F| = 2^{2^n} = 2^N$. Let $S \subseteq F$ be the subset of these functions which are computed by circuits of size $2^n/(30n)$. Then we know that $|S|$ is at most $2^{2^n/10}$. For any function $f \in F$, let $T(f)$ denote the truth table of f. For any $s \in S$, consider a hamming ball of radius $N/4$ around $T(s)$. The hamming ball contains all $T(f)$ such that s computes f on at least $3/4$ fraction of inputs. Since $|S|$ is at most $2^{2^n/10}$ and the volume of a hamming ball (from previous Lemma) is at most $2^{0.85N}$, the number of functions which are computable on at least $3/4$ fraction of inputs by at least one of the functions $\in S$ is at most $2^{0.95N}$. This implies that there is a function f such that f is $(1/4)$-hard for circuits of size $2^n/(30n)$.

□

Corollary I.B.7. *Let $n \in \mathbb{N}$. There exists a boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that f is $(1/4)$-hard for circuits of size $\text{poly}(n)$ for any polynomial poly and all sufficiently large values of n.*

Finally, we state the direct product theorem with respect to boolean function against bounded size circuits.

Theorem I.B.8. *Let $n, k, s \in \mathbb{N}$, $0 < \delta \leq 1$, and $f : \{0,1\}^n \rightarrow \{0,1\}$ be a boolean function. Define $f^k : \{0,1\}^{nk} \rightarrow \{0,1\}^k$ as $f^k(x_1, ..., x_k) \overset{def}{=} f(x_1).f(x_2)...f(x_k)$.*

If f is δ-hard for circuits of size s, then f^k is $(1-\epsilon)$-hard for circuits of size s', where $\delta = \Theta\left(\frac{\log 1/\epsilon}{k}\right)$ and $s' = s \cdot \mathrm{poly}(\delta, \epsilon, 1/k, 1/n)$ for some polynomial poly.

A number of proofs of the above theorem can be found in the literature. We will discuss them in Section I.C. Let us try to see the intuition behind the above Theorem. Consider a biased coin which gives 1 with probability $(1-\delta)$ and 0 with probability δ whenever it is tossed. The coin toss turning 1 can be associated to the function being computable on a randomly chosen input and similarly for the coin toss turning 0. Then the probability that k independent coin tosses all turning 1 is $(1-\delta)^k$. So, intuitively the probability that a circuit simultaneously computes the function on k randomly chosen inputs should also be proportional to $(1-\delta)^k$. Even though this fact is true, the argument is not so simple. The flaw in the argument using the coin toss analogy is that a circuit computing f^k is not constrained to compute different instances of the function independently. We have to account for any other strategy that the circuit might possibly employ to compute f^k. This motivates proof by contradiction where we start with the assumption that there is a circuit which computes f^k with probability at least $(1-\delta)^k$ and then use it to construct a circuit which computes f with probability at least $(1-\delta)$. We defer this discussion until Section I.C.

In the next subsection we take a look at the XOR Lemma which came up earlier in our discussion and which has applications in Cryptography and Error-correcting codes. Here we will see how the XOR Lemma follows from the direct product theorem and vice versa.

I.B.1 Relationship with XOR Lemma

Following is the classical XOR Lemma.

Theorem I.B.9 (XOR Lemma)**.** *Let $n, k, s \in \mathbb{N}$, $0 < \delta \leq 1$, and $f : \{0,1\}^n \rightarrow \{0,1\}$ be a boolean function. Define $f^{\oplus k} : \{0,1\}^{nk} \rightarrow \{0,1\}$ as $f^{\oplus k}(x_1, ..., x_k) \overset{def}{=} f(x_1) \oplus f(x_2) \oplus ... \oplus f(x_k)$. If f is δ-hard for circuits of size s, then $f^{\oplus k}$ is $(1/2-\epsilon)$-*

hard for circuits of size s', where $\delta = \Theta\left(\frac{\log(1/\epsilon)}{k}\right)$ and $s' = s \cdot \mathrm{poly}(\epsilon, \delta, 1/k, 1/n)$ for some polynomial poly*.*

The proof of the above XOR Lemma follows from the direct product theorem I.B.8. The proof requires the following theorem due to Goldreich and Levin.

Theorem I.B.10 ([GL89])**.** *Let $x \in \{0,1\}^n$ be any string, $s' \in \mathbb{N}$ and let C' be a boolean circuit on n variables of size s' such that $\mathbf{Pr}_{r\in\{0,1\}^n}[C'(r) = \langle x, r\rangle] \geq 1/2 + \gamma$, for some $\gamma > 0$. Then, there is a randomized circuit[6] C of size $s = s' \cdot \mathrm{poly}(n, 1/\gamma)$ such that C outputs the string x with probability at least $\Omega(\gamma^2)$.*

Proof of Theorem I.B.9 using Theorem I.B.8. For the sake of contradiction, assume that there is a circuit C such that C computes $f^{\oplus k}$ with probability at least $1/2+\epsilon$. We will show a construction of a circuit which computes the function f with probability at least $(1-\delta)$.

Consider the function $F : \{0,1\}^{2nk} \times \{0,1\}^{2k} \to \{0,1\}$ defined as follows: For $x_1, \ldots, x_{2k} \in \{0,1\}^n$ and $r \in \{0,1\}^{2k}$,

$$F(x_1, \ldots, x_{2k}, r) = \langle f(x_1) \ldots f(x_{2k}), r\rangle.$$

Note that conditioned on $r \in \{0,1\}^{2k}$ having exactly k 1s, the function $F(x_1, \ldots, x_{2k}, r)$ is distributed exactly like the function $f^{\oplus k}(x_1, \ldots, x_k)$, for uniformly and independently chosen x_is.

Consider the following circuit for computing F. Given an input $x_1, \ldots, x_{2k}, r$, count the number of 1s in the string r. If it is not equal to k, then output a random coin flip and halt. Otherwise, simulate the circuit C on the sub-tuple of $x_1, \ldots, x_{2k}$ of size k which is obtained by restricting $x_1, \ldots, x_{2k}$ to the positions in r that are 1, and output the answer of C.

Let p be the probability that a random $2k$-bit string contains exactly k 1s. It is easy to see that the described circuit for computing F is correct with

[6] *a randomized circuit is a circuit that also takes as input a random string.*

probability at least $(1-p)/2 + p(1/2+\epsilon) = 1/2 + p\epsilon$. Since $p \geq \Omega(1/\sqrt{k})$, we get that our circuit for F is correct with probability at least $1/2+\epsilon'$, for $\epsilon' = \Omega(\epsilon/\sqrt{k})$.

By a Markov-style argument[7], we have that for each of at least $\epsilon'' = \epsilon'/2$ of the $2k$-tuples $x_1, \ldots, x_{2k}$, our circuit computes $F(x_1, \ldots, x_{2k}, r)$ for at least $1/2+\epsilon''$ fraction of rs. Using the Goldreich-Levin circuit of Theorem VI.B.22, we get a randomized circuit that computes $f^{2k}(x_1, \ldots, x_{2k})$ with probability at least $\epsilon''' = \Omega(\epsilon''^3)$, where the probability is both over the input $2k$-tuples $x_1, \ldots, x_{2k}$ and the internal randomness of our randomized algorithm. By averaging, randomly fixing the internal randomness of the algorithm yields, with probability at least $\epsilon'''/2$, a deterministic circuit $(\epsilon'''/2)$-computing the direct product function $f^{2k}(x_1, \ldots, x_{2k})$. Finally, applying Theorem I.B.8 to this direct product circuit yields, with probability poly(ϵ), a circuit computing f on at least $1-\delta$ fraction of inputs of f, as required. $\square$

We proved the XOR Lemma using the direct product theorem. We can also prove the direct product theorem using the XOR Lemma thus showing that both these hardness amplification results are essentially saying the same thing.

Theorem I.B.11 (XOR Lemma implies Direct Product Theorem)**.** *Let $n, k, s \in \mathbb{N}$, $\epsilon \geq 4 \cdot e^{-k/4}$, and $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. If $\forall k' \in [k/4, 7k/4], f^{\oplus k'}$ is $(1/2 - \epsilon/4)$-hard for circuits of size s, then f^{2k} is ϵ-hard for circuits of size $s' = s$.*

Proof. For the sake of contradiction, assume that there is a circuit C of size at most s' such that $\mathbf{Pr}_{(x_1,\ldots,x_{2k})}[C(x_1,...,x_{2k}) = f^k(x_1,...,x_{2k})] \geq \epsilon$. Consider the following circuit which attempts to compute $\langle f^{2k}(.), r\rangle$ for a randomly chosen $r \in \{0,1\}^{2k}$:

> C': "Given an input $((x_1,...,x_{2k}), r)$, where $r \in \{0,1\}^{2k}$ output $\langle C(x_1,...,x_{2k}), r\rangle$"

We have:

$$\mathbf{Pr}_{(x_1,\ldots,x_{2k}),r}[C'((x_1,...,x_{2k}), r) = \langle f^{2k}(x_1,...,x_{2k}), r\rangle] \geq 1/2 + \epsilon/2 \qquad \text{(I.1)}$$

[7]for random variables X, Y, event $E(X,Y)$ and for every $\alpha \in [0,1]$, if $\mathbf{Pr}_{X,Y}[E(X,Y)] \geq \alpha$, then $\mathbf{Pr}_X[\mathbf{Pr}_Y[E(X,Y)] \geq \alpha/2] \geq \alpha/2$

The probability that the number of 1's in a random $2k$-bit string is not in the range $[k/4, 7k/4]$ is at most $e^{-k/4}$ from Chernoff bounds. This is at most $\epsilon/4$ from the assumed bound on ϵ. Combining this fact with equation I.1 we get that there exists a $k' \in [k/4, 7k/4]$ such that

$$\mathbf{Pr}_{(x_1,...,x_{2k}),r,|r|=k'}[C'((x_1,...,x_{2k}),r) = \langle f^{2k}(x_1,...,x_{2k}),r\rangle] \geq 1/2 + \epsilon/4$$

This implies that the following circuit computes the function $f^{\oplus k'}$ with probability at least $1/2 + \epsilon/4$

> C'': On input $(x_1, ..., x_{k'})$, construct an input $(y_1, ..., y_{2k})$ by choosing $2k - k'$ inputs randomly and randomly arranging the chosen inputs and the given inputs. Finally, C'' evaluates $C(y_1, ..., y_{2k})$ and outputs the xor of bits at position where the given inputs are placed within $y_1, ..., y_{2k}$.

□

I.C Direct Product Theorems: A History

A seminal paper by Yao [Yao82] introduced a number of ideas in Complexity Theory. One of them was Hardness Amplification which is the general idea that multiple instances of a problem is harder to solve than a single instance. More specifically, a version of the XOR Lemma I.B.9 was informally introduced without a formal proof. The first formal proof of the XOR Lemma was given by Levin in [Lev87]. Due to its applicability in areas of Complexity Theory and Cryptography various other proofs of the XOR Lemma and the direct product theorem were given. Each proof engineered for a different application. The formal connection between the direct product theorem and the XOR Lemma, using the Goldreich-Levin result was given in a very nice survey [GNW95] on the topic by Goldreich, Nisan, and Wigderson. Next, we look at the three main styles of proof arguments that have been used for proving the direct product theorem. We do not give a formal proofs though.

I.C.1 Levin-style Argument

Levin [Lev87] gave the first proof of the XOR Lemma which was later expanded on in a survey work [GNW95] on this topic. The presentation in this subsection follows the presentation in [GNW95].

Here we will sketch the proof of the direct product theorem for the special case of $k = 2$. This will be sufficient to give the intuition of the main ideas involved in the proof. Informally, we will show that if an arbitrary function $f : \{0,1\}^n \rightarrow R$ (R is the range of the function) is δ-hard for circuits of size s, then the function defined as $f^2(x_1, x_2) \overset{def}{=} f(x_1).f(x_2)$ is $(1-(1-\delta)^2-\epsilon)$-hard for circuits of size $s \cdot \frac{\epsilon^2}{n}$. The arguments can be extended for arbitrary k using induction. Note that this proof argument works for arbitrary functions and is not restricted to boolean functions.

The proof, as is the case with direct product theorems, is by contradiction. So, let us assume that there is a circuit C such that computes f^2 with probability at least $(1-\delta)^2 + \epsilon$. Let X, Y denote independent random variables uniformly distributed over $\{0,1\}^n$. Then we have

$$\begin{aligned} \mathbf{Pr}[C(X,Y) = f^2(X,Y)] &= \mathbf{Pr}[C(X,Y) = f(X).f(Y)] \\ &\geq (1-\delta)^2 + \epsilon. \end{aligned} \tag{I.2}$$

We will show that we can use this circuit to either construct a circuit C' which computes f well in average. Let us write the success probability of C as

$$\begin{aligned} \mathbf{Pr}[C(X,Y) = f(X).f(Y)] &= \mathbf{Pr}[C(X,Y)[2] = f(Y)] \\ &\quad \cdot \mathbf{Pr}[C(X,Y)[1] = f(X) | C(X,Y)[2] = f(Y)] \end{aligned}$$

where $C(X,Y)[i], i \in \{1,2\}$ denotes the output of C corresponding to the i^{th} instance of the function. Next, we need to consider the following two cases:

1. In the case $\mathbf{Pr}[C(x,Y)[2] = f(Y)] > (1-\delta)$ for any fixed $x \in \{0,1\}^n$, then we can construct the following circuit C' for computing f:

> "Given an input $y \in \{0,1\}^n$, output $C(x,y)[2]$."

Note that $\mathbf{Pr}[C'(Y) = f(Y)] > (1-\delta)$.

2. In case $\mathbf{Pr}[C(X,Y)[1] = f(X) | C(X,Y)[2] = f(Y)] > (1-\delta) + \epsilon$, we can construct the following circuit C_2: Suppose we are given a set S of random samples of the function f, that is, S contains pairs of the form $(y, f(y))$ for random $y \in \{0,1\}^n$. Then consider the following circuit C' for computing f:

   > "Given an input x, let S_C denote the subset of samples in S such that for any $(y, f(y)) \in S_C$, $C(x,y)[2] = f(y)$. Pick a random element $(y, f(y))$ from the set S_C and output $C(x,y)[1]$."

   Given that $|S| = \Omega(n/\epsilon^2)$, we have

$$\begin{aligned}\mathbf{Pr}_x[C'(x) = f(x)] &= \mathbf{Pr}_{x,(y,f(y))\in S_C}[C(x,y)[1] = f(x) | C(x,y)[2] = f(y)] \\ &\geq \mathbf{Pr}[C(X,Y)[1] = f(X) | C(X,Y)[2] = f(Y)] - \epsilon \\ &> (1-\delta).\end{aligned}$$

Finally, given that $\mathbf{Pr}[C(X,Y) = f(X).f(Y)] > (1-\delta)^2 + \epsilon$, either case (1) or case (2) holds. This means that there is a circuit C' which computes f on at least $(1-\delta)$ fraction of the inputs.

There are some important issues which remain unaddressed in the above discussion . For example, the above proof argument is non-constructive, that is, we only showed the existence of a circuit C' which computes f well on average. So the question is whether we can give an explicit construction of such a circuit C' by doing a more careful analysis? We assumed that we are given a samples from the function f, that is, pairs $(y, f(y))$ for randomly chosen y's. Since we are talking about circuits, we can assume that these samples are hardwired into the circuit computing f. However, this seems to be another bottleneck when trying to come up with a constructive proof. So, an important question is whether such samples are necessary for proving the direct product theorem and if so, can we quantify this? We will answer this question in Chapter II.

I.C.2 Impagliazzo's Hard-core Set Argument

A fundamental result by Impagliazzo [Imp95a] reshaped and increased our understanding about hardness of boolean functions. As defined before, a boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is δ-hard for circuits of size s if no circuit of size s can compute the function on at least $(1-\delta)$ fraction of the inputs. So, any fixed circuit makes mistakes on at least δ fraction of the inputs. The natural question to ask is whether circuits of size s makes mistakes on different sets of $\delta \cdot 2^n$ inputs, or is there a hard set of inputs where the mistakes are of all such circuits are restricted. Intuitively, if different circuits make mistakes on very different sets in inputs, then we might be able to combine these circuits to construct another circuit which computes f on almost all inputs. [Imp95a] showed that this intuition is indeed correct and there indeed exists a "hard-core" set of inputs. [Hol05] improved the results of [Imp95a] to obtain optimal size of such a hard-core set. Here the statement of the theorem.

Theorem I.C.1 ([Imp95a] and [Hol05])**.** *Let $n, s \in \mathbb{N}$, $0 < \delta, \epsilon \leq 1$, and $f : \{0,1\}^n \rightarrow \{0,1\}$ be a boolean function. If f is δ-hard for circuits of size s, then there is a set $H \subseteq \{0,1\}^n$ such that $|H| \geq (2\delta)2^n$ and for any circuit C of size at most $d\epsilon^2\delta^2 s$, we have*

$$\mathbf{Pr}_{x \in H}[C(x) = f(x)] \leq 1/2 + \epsilon/2,$$

where d is an absolute constant.

The above theorem essentially says that if a function is δ-hard for circuits of size s, then there is a subset H of inputs of density at least 2δ such that no circuit of certain smaller size can compute the function much better than randomly guessing the function value on inputs in the subset. The set H is usually referred to as the hard-core set for the function f.

The above theorem can be used to prove the XOR Lemma in the following manner: Since f is δ-hard, from the above theorem we get that there is a subset

$H, |H| \geq (2\delta)2^n$ of hard inputs. For the sake of contradiction, assume that there is a circuit C which computes $f^{\oplus k}$ with probability at least $(1-2\delta)^k + \epsilon k/2$. Let $A_l, l \geq 0$ denote the set of inputs $(x_1, ..., x_k)$ such that exactly l, x_i's are from the hard-core set H. Since $|A_0| \leq (1-2\delta)^k$, there exists an $l \geq 1$ such that

$$\mathbf{Pr}_{(x_1,...,x_k)\in A_l}[C(x_1, ..., x_k) = f^{\oplus k}(x_1, ..., x_k)] \geq 1/2 + \epsilon/2. \tag{I.3}$$

Using this, we can construct the following circuit C' for computing f:

> "On input x, chooses $l-1$ random inputs $(a_1, ..., a_{l-1})$ from the hard-core set H, $k-l-1$ inputs $(b_1, ..., b_{k-l-1})$ from the complement of H, and then constructs an input for the circuit C by randomly arranging these inputs. If $f(a_1) \oplus ... \oplus f(a_l) \oplus f(b_1) \oplus ... \oplus f(b_{k-l-1}) = 0$ then return the value of C evaluated on the constructed input, otherwise return the complement answer."

From (I.3) we get that $\mathbf{Pr}_{x\in H}[C'(x) = f(x)] \geq 1/2 + \epsilon/2$ which contradicts with the fact that H is the hard-core set for the function f. We can obtain the direct product theorem by using the reductions of the previous section.

We should note certain things about the above discussion. Firstly, this proof argument works only for boolean functions. Secondly, we get a weaker hardness parameter (weaker by a factor k) compared to the Levin-style argument. Finally, as in the previous section, we need samples of the function f which again raises the question whether such samples are necessary for the proof.

I.C.3 Trust Halving Strategy

This proof method called the "trust halving strategy" was introduced in [IW97]. This was motivated by research in Derandomization that we discussed in one of the previous sections. There was a need for a derandomized version of the XOR Lemma[8] and new techniques which allowed the possibility of derandomization were needed. We look at one such techniques in this subsection.

It will be easier to talk in terms of advantage of a circuit which is defined as below:

[8]An XOR Lemma where the inputs are not chosen independently and might have some dependencies

Definition I.C.2. Let $n \in \mathbb{N}$ and $f : \{0,1\}^n \rightarrow \{0,1\}$ be any boolean function. Advantage of a circuit C over a function f on a set of inputs S is defined as

$$Adv(C,f) = \mathbf{Pr}_{x \in S}[C(x) = f(x)] - \mathbf{Pr}_{x \in S}[C(x) \neq f(x)]$$

This is related to the success probability of the circuit in guessing f on the set of inputs by the following simple fact.

Fact I.C.3. Let $n \in \mathbb{N}$ and $f : \{0,1\}^n \rightarrow \{0,1\}$ be any boolean function. A circuit C has advantage ϵ over function f on a set of inputs S iff $\mathbf{Pr}_{x \in S}[C(x) = f(x)] = 1/2 + \epsilon/2$.

We will again use Impagliazzo's hard-core theorem of the previous subsection. Given a function which is δ-hard , we know that there is a subset H of hard-core inputs such that $|H| \geq 2\delta 2^n$ and any small circuit has negligible advantage on the inputs contained in H. Towards contradiction, we will start with the assumption that there is a circuit which computes the direct product function f^k on non-negligible fraction of inputs and then construct a circuit which has non-negligible advantage over the hard-core set of inputs H.

Here is the description of this circuit C':

> "Given an input $x \in H$, the circuit randomly selects inputs $x_1, ..., x_{k-1} \in \{0,1\}^n$, an index $i \in \{1...k\}$, constructs an input $(y_1, ..., y_k) = (x_1, ..., x_{i-1}, x, x_{i+1}, ..., x_{k-1})$ and execute C on this input. Let $T(y_1, ..., y_k)$ be the set of indices other than i where C makes a mistake given input $(y_1, ..., y_k)$, that is, $T(y_1, ..., y_k) = \{j : C(y_1, ..., y_k)[j] \neq f(y_j), j \neq i\}$. C' makes a "soft" decision about outputting $C(y_1, ..., y_k)[i]$ depending on the size of the set T. In some sense, C' has more "trust" in C if C is giving the correct answers on the indices other than i. Once again, C' can compute L since we assume that it has access to random samples of the function f which are hardwired into C'. C' outputs $C(y_1, ..., y_k)[i]$ with probability $2^{-|T(y_1,...,y_k)|}$ and with the remaining probability it outputs an arbitrary answer."

Let us now analyze the advantage of C' over f on H. First, note that constructing an input $(y_1, ..., y_k)$ in order to query C induces a distribution on the k-tuple of inputs which is different from the uniform distribution on $\{0,1\}^{nk}$.

More specifically, for any fixed $(y_1, ..., y_k)$, let $L(y_1, ..., y_k)$ denote the set of indices such such that C evaluated on $(y_1, ..., y_k)$ makes mistakes on these indices, that is, $L(y_1, ..., y_k) = \{i : C(y_1, ..., y_k)[i] \neq f(y_i), i \in \{1, ..., k\}\}$. Then the probability that a k-tuple $(y_1, ..., y_k)$ is sampled is $\frac{|L(y_1,...,y_k)|}{2\delta k}$ times the probability of sampling from uniform distribution. This is because i can be chosen as any one of these s positions, and once i is fixed, we have 1 out of $2\delta 2^n$ ways to fix y_i and 1 out of 2^n ways to fix the other inputs. For further analysis, let us divide the k-tuples into the following subsets. We will evaluate the advantage of C' conditioned on sampling a k-tuple from these subsets.

1. $\{(y_1, ..., y_k)||L(y_1, ..., y_k)| < \delta k\}$: From a simple application of Chernoff bounds we know that the density of this subset of tuples is at most $e^{-\delta k/4} < \epsilon/8$. So, in the worst case this subset of tuples contribute at least $-\epsilon/8$ to the advantage.

2. $\{(y_1, ..., y_k)||T(y_1, ..., y_k)| = 0, |L(y_1, ..., y_k)| \geq \delta k\}$: The conditional advantage over this set of tuples is 1. Furthermore, the density of this subset of tuples is at least $\epsilon - \epsilon/4$. So it contributes at least $(\epsilon - \epsilon/4) \cdot \frac{\delta k}{2\delta k} \cdot 1 = 3\epsilon/8$ to the total advantage.

3. $\{(y_1, ..., y_k)|0 < |T(y_1, ..., y_k)| \leq \delta k/4, |L(y_1, ..., y_k)| \geq \delta k\}$: For a fixed $(y_1, ..., y_k)$ in this set, the conditional advantage can be written down as
$$\left(1 - \frac{|T(y_1, ..., y_k)|}{|L(y_1, ..., y_k)|}\right) 2^{-|T(y_1,...,y_k)|} - \left(\frac{|T(y_1, ..., y_k)|}{|L(y_1, ..., y_k)|}\right) \cdot 2^{-|T(y_1,...,y_k)|+1}$$
which is a positive quantity even in the worst case.

4. $\{(y_1, ..., y_k)||T(y_1, ..., y_k)| > \delta k/4\}$: In the worst case the conditional advantage is $-2^{-\delta k/4}$. So this subset contributes at least $-\epsilon/8$ to the total advantage.

So, the total advantage is at least $(3\epsilon/8 - \epsilon/8 - \epsilon/8) = \epsilon/8$ which then contradicts with the hard-core Theorem.

Let us now see some of the important things one should note about the above proof argument. The crucial idea used in the above proof argument is the following property:

> (*) For any subset of inputs $S \subseteq \{0,1\}^n$ of density at least 2δ, only negligible fraction of k-tuples $(x_1, ..., x_k)$ (inputs for f^k) contain less than δk inputs from S.

Given this, the proof argument can be summarized as follows: we associate the set S with the hard-core set for the function f. Then almost all the inputs on which C gives correct answers have lots of inputs from the hard-core set. This makes it possible to construct a circuit which achieves good advantage over the hard-core set given that C answers correctly on some non-negligible fraction of k-tuples.

The property above can be easily shown using Chernoff bounds given that the k-tuple is chosen independently. If we can show a similar property when the inputs are not chosen independently but might have some dependencies (and hence a shorter description), then we almost immediately obtain a derandomized direct product theorem. [IW97] consider k-tuple of inputs obtained by doing a random walk on certain small degree graphs. The property is shown to be true and a derandomized direct product theorem is obtained.

II

Direct Product Theorems: A Coding Theoretic Perspective

(Parts of this Chapter is based on a joint work with Russell Impagliazzo and Valentine Kabanets.)

II.A Introduction

In the previous Chapter, we looked at direct product theorems under a simple setting. We considered circuits computing functions with fixed size inputs. In this Chapter, we look at more general settings and study the difficulties that arise in such generalizations.

We will consider general functions $f : \{0,1\}^* \to \{0,1\}$ and Languages. In order to study the hardness of general functions and languages with respect to circuits we will need to consider circuit *family* which is defined in the following manner.

Definition II.A.1 (Circuit Family)**.** Let $s : \mathbb{N} \to \mathbb{N}$ be a function. An $s(n)$-size circuit family is a sequence $\{C_n\}_{n\in\mathbb{N}}$ of boolean circuits such that C_n has n inputs and one output and for every n, $|C_n| \leq s(n)$.

We can now define hardness of general boolean functions with respect to circuit families.

Definition II.A.2 (Average-case hardness of geneal functions wrt. circuit families)**.** Let $s : \mathbb{N} \to \mathbb{N}$, $\delta : \mathbb{N} \to \mathbb{N}$, and $f : \{0,1\}^* \to \{0,1\}$ be functions. f is called $\delta(n)$-hard for size $s(n)$-size circuit families if for sufficiently large n, any boolean circuit C on n inputs and of size at most $s(n)$ satisfies

$$\mathbf{Pr}_{x \in \{0,1\}^n}[C(x) = f(x)] \leq (1 - \delta(n)).$$

In most parts of the following discussion for general boolean functions, we will implicitly assume that δ and s are functions and simply say that f is δ for size s.

We will also need to talk about average-case hardness of languages within some complexity class. So, let us define what we mean by a language being decidable by circuit families.

Definition II.A.3 (Language Recognition by Circuit Families)**.** Let $s : \mathbb{N} \to \mathbb{N}$ be a function. We say that a language L is decidable by $s(n)$-size circuit families or simply L is in size $s(n)$, if there is an $s(n)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$ such that for all sufficiently large n, for every $x \in \{0,1\}^n$, $x \in L$ iff $C_n(x) = 1$.

Note that there is a very crucial difference between a language being decidable by circuit families compared to a language being decidable by Turing machines. Apart from the fact that model of computation is different, note that for the case of circuits we are allowing a different circuit for each problem size. On the other hand, for Turing machines we are only allowed a single Turing machine for all problem sizes. It is not immediately clear as to which definition is stronger. These are *uniform* (Turing machines) and *nonuniform* (circuits) models of computation and we discuss them in the next subsection.

II.A.1 Uniform versus Nonuniform Computation

The uniform model of computation allows a single machine for instances of all sizes for a given problem. It is synonymous with Turing machines. The resource we are interested here is the running time of the Turing machine.

On the other hand, nonuniform model of computation allows a different machine for instances of different sizes for a given problem. An example of a nonuniform model of computation is a circuit family. The resource we are interested in is the size of this circuit family. Another example is a Turing machine which is allowed an advice string as an additional input which only depends on the size of the input. The resource which are of interest in this case is the running time of the Turing machine and the size of the advice as a function of the instance size of the problem. Intuitively, this model seems to be more powerful than Turing machines since there is an additional advice string which could assist in solving problems. An important question then is "in what ways is this a stronger model of computation compared to the uniform model?". One way to answer this question is to define a complexity class for this model and then study the relation of this complexity class with analogous classes defined in the uniform model. Let $\mathsf{P}/poly$ denote the class of problems decidable by polynomial time Turing machines which are allowed polynomial amount of advice. Alternatively, this is the class of problems which are decidable by polynomial size circuit families. This class contains P (the class of problems which can be solved using polynomial time Turing machines) but under some strongly believed complexity theoretic assumptions $\mathsf{NP} \not\subseteq \mathsf{P}/poly$. Furthermore, it is hard to upper bound the computational power of this class in general as this class also contains some undecidable languages.

In this Dissertation, we will consider a "semi-uniform" model of computation following [TV02]. Here we define the class $\mathsf{BPP}//log$ as the class of probabilistic algorithms with advice of length O(log n) that depends on the random coin tosses of the algorithm, but not on the input. We can view such an algorithm as producing a polynomial-sized list of polynomial-size circuits

Direct product theorems provide another scenario which brings out the gap between the computational power of nonuniform/semi-uniform and uniform model of computation. We will study this difference in the following section.

II.B Why Direct Product Theorems Fail in a purely Uniform Setting?

In the previous Chapter we considered average-case hardness of functions with fixed input size against bounded size circuits. We generalized this to hardness of general functions against bounded size circuit families. In this section, we consider uniform model of computation. We look at avergae-case hardness of general functions against Turing machines (algorithms) with bounded running time. We will see that notion of hardness within the uniform model is weaker than the analogous notion of hardness in the nonuniform model. This is in the sense that strong amplification of hardness within the uniform model using direct product theorems is not possible. On the other hand, we have seen direct product theorems in the nonuniform model. First we need to define average-case hardness of general functions in the uniform model.

Definition II.B.1 (Average-case hardness in the uniform model)**.** Let $t : \mathbb{N} \to \mathbb{N}$, $\delta : \mathbb{N} \to \mathbb{N}$, and $f : \{0,1\}^* \to \{0,1\}$ be functions. f is said to be $\delta(n)$-hard for time $t(n)$ if for all sufficiently large n and every algorithm[1] A with running time $t(n)$, we have

$$\mathbf{Pr}_{x \in \{0,1\}^n}[A(x) = f(x)] < (1 - \delta(n)).$$

The probability is over the choice of inputs of a given length and the internal randomness of the algorithm A.

We now show that a string direct product theorem in the uniform setting is not possible.

Theorem II.B.2. *There is a function $f : \{0,1\}^* \to \{0,1\}$ such that f is $(1/5)$-hard for polynomial time and there is a polynomial time randomized algorithm A such that for all sufficiently large n*

$$\mathbf{Pr}_{(x_1,...,x_k) \in \{0,1\}^{nk}}[A(x_1, ..., x_n) = f^k(x_1, ..., x_n)] \geq 1/n.$$

[1]the algorithm can be randomized, that is, it is allowed to use some randomness in computation.

Proof. Consider the following set F_n of functions on n bit inputs: the functions in F_n depend on only the first $\log n$ bits of the inputs. The functions in F_n can be indexed by a string $x \in \{0,1\}^{\log n}$ and let this function be denoted by $f_{n,x}$. For any $x \in \{0,1\}^{\log n}$, $f_{n,x}$ is defined as:

$$\forall y \in \{0,1\}^n, \quad f_{n,x}(y) \overset{def}{=} \langle x, y[1]...y[\log n] \rangle$$

Clearly, $\forall n, |F_n| = n$. Note that for any pair of functions $(f_{n,x}, f_{n,y})$, $f_{n,x}, f_{n,y} \in F_n, x \neq y$ we have

$$\Delta(T(f_{n,x}), T(f_{n,y})) \geq (1/2) \cdot 2^n$$

where T denotes the truth table of a function and Δ denotes the hamming distance between two strings. Consider probabilistic Turing machines which have description length at most $(\log n - 1)$. Note that there are $n-1$ such machines. The following lemma shows that there is a function $\in F_n$ which is hard on average for all these machines.

Claim II.B.3. *There is a function $f \in F_n$ such that for any probabilistic Turing machine M which has description length at most $(\log n - 1)$ we have*

$$\mathbf{Pr}_{x \in \{0,1\}^n}[M(x) = f(x)] \leq 4/5.$$

Proof. Consider hamming balls of radius $2^n/5$ centered around $T(f)$ for all functions $f \in F_n$. Since the minimum distance between the truth table of any two functions in F_n is at least $2^n/2$, these hamming balls do not intersect. Let $T(M)$ denote the truth table of a machine . Since there are only $n-1$ probabilistic Turing machines with description size at most $(\log n - 1)$, there is at least one hamming ball which does not contain the truth table of any function g which is computable by any of these $(n-1)$ machines. This means that the function f corresponding to this hamming ball is the hard function which satisfies the claim in the lemma. □

In the previous claim, we have essentially showed that there is a function $f : \{0,1\}^* \rightarrow \{0,1\}$ such that for every probabilistic Turing machine M, there is an $n_0 \in \mathbb{N}$ such that $\forall n \geq n_0, \mathbf{Pr}_{x \in \{0,1\}^n}[M(x) = f(x)] \leq 4/5$.

Let us now prove the second part of the theorem. For any function $f : \{0,1\}^* \rightarrow \{0,1\}$ defined in the manner above there is a simple probabilistic Turing machine M which solves f^k.

> "Given any input $(x_1, ..., x_k)$ such that $|(x_1, ..., x_k)| = nk$, M chooses a a function $f \in F_n$ uniformly at random and returns $(f(x_1), ..., f(x_k))$."

Since $|F_n| = n$, we get the probability that M succeeds is at least $1/n$. This proves the theorem. □

The theorem above shows that there is a function for which we cannot amplify the hardness beyond a polynomial amount.

II.C Direct Product Codes

In the previous section, we showed that an impossibility result for strong direct product theorems in the Uniform setting. In this section, we will analyze the reasons behind this. We will study a related error-correcting code called the direct product code and analyze its properties. Some interesting properties of this code will give us valuable intuition behind the impossibility of strong direct product theorems in the uniform setting.

Definition II.C.1 (Direct Product Code)**.** Let $n, k \in \mathbb{N}$, $N = 2^n$ and $\Sigma = \{0,1\}^k$. A k-wise direct product Code $C : \{0,1\}^N \rightarrow \Sigma^{N^k}$ is defined as follows: Given a message $m \in \{0,1\}^N$ in binary, the encoding of the message $C(m)$ is over the alphabet Σ. Let the message be indexed by strings in $\{0,1\}^n$, where the bit indexed by $x \in \{0,1\}^n$ is denoted by $m[x]$. The codeword is indexed by a k-tuple of strings in $\{0,1\}^n$. A letter in the codeword indexed by $(x_1, ..., x_k) \in \{0,1\}^{nk}$ is denoted by $C(m)[(x_1, ..., x_k)]$ and is defined as

$$C(m)[(x_1, ..., x_k)] = m[x_1].m[x_2]...m[x_k]$$

Note the similarity of the k-wise direct product code with the direct product theorem. Given a function $f : \{0,1\}^n \rightarrow \{0,1\}$, if we consider the truth table of this function as a message for the encoding then the truth table of the direct product function f^k is essentially the encoding of the message. Let us now see how we can make use of this similarity. Suppose the proof of the direct product theorem was by contradiction, that is, we start with the assumption that there is a circuit which computes f^k on some non-negligible number of inputs and then constructed another circuit which computes f on almost all inputs. Such a proof can be interpreted as decoding algorithm for the k-wise direct product code. This is because we essentially give a procedure to construct (a circuit which computes) a string that is close to the message, given (a circuit which computes) a corrupted codeword.

In the other direction, the decoding properties of this code gives important knowledge about the limitation of direct product theorems. In the next subsection we study some of these properties of the direct product codes.

II.C.1 List Decoding Direct Product Codes

Note that the k-wise direct product code has small distance since any two messages which are close[2] to each other have almost the same codewords. To make things worse, for our purposes, we would like to reconstruct a message given its corrupted codeword which is corrupted almost everywhere. In an ideal setting, we would like to give a decoding algorithm which reconstructs a message uniquely from a corrupted codeword. However, this would not be possible for direct product codes. So we relax the notion of decoding in the following ways:

1. *List decoding*: We would like to decode starting from an extremely corrupted codeword which can be corrupted in at all but ϵ fraction of positions for small ϵ. Intuitively, there would be lots of messages such that their codewords are at a distance at most $(1-\epsilon)$ from the given corrupted codeword. Since any of these messages could be the original message, we have to allow the decoding algorithm to output a list of messages instead of a single message.

[2]here we mean close in the hamming sense

2. *Approximate decoding*: Note that since we have a small distance code, the list of messages above could be very large. We can reduce the list size for the same amount of measure of corruption if we further relax the notion of decoding. We allow the decoding algorithm to output a list of messages such that at least one of the messages is close to the original message.

3. *Local decoding*: Finally, we are interested in these code from the perspective of direct product theorems. In direct product theorems we are interested in constructing a circuit which when given an input (given a position in the message), queries the circuit for the direct product function (reads some positions of the corrupted codeword) and outputs the function value (bit at the given position) of the input. This corresponds an idea called *local decoding* which has been used in the error-correcting code literature to allow sub-linear time (sub-linear in the code size) decoding algorithms.

So we would be interested in local, approximate, list decoding of direct product codes. With that motivation, let us now study the combinatorial properties of direct product codes which would be of interest to us.

Definition II.C.2 (Hamming distance)**.** Let $k \in \mathbb{N}$ and Σ be some alphabet. Given two string $x, y \in \Sigma^k$, the hamming distance between x and y which we denote by $\Delta(x, y)$ is defined as

$$\Delta(x, y) = |\{i : x[i] \neq y[i]\}|$$

Approximate list decoding requires us to output the smallest possible list of messages such that at least one of them is close to the original message with closeness parameter δ. More specifically, there should be at least one element in the list which is agrees with the original message on at least $(1 - \delta)$ fraction of positions. Given this, we can assume that the list does not contain a pair of messages which disagree on less than δ fraction of positions (if it does then we can make the list smaller by removing a message from the list). We first show that for fixed $\delta \leq 1/17$, if ϵ is very small, then there could be exponentially many messages in the list such that their codewords agrees with the given word on at least ϵ fraction of positions and each pair of messages in the list disagree on at least δ fraction of the positions. Consider the following theorems

Theorem II.C.3. *Let $\delta \leq \min\{(\ln 1/\epsilon)/(2k), 1/17\}$. There exists a set of at least $2^{\delta N-1}$ of N-bit messages of pairwise Hamming distance at least δN and such that the direct product encoding of each message has agreement at least ϵ with the string 0^{N^k}.*

The proof of this theorem will follow from the two lemmas below.

Lemma II.C.4. *Let $\delta \leq \min\{(\ln 1/\epsilon)/(2k), 2/3\}$ and let $m_1, \ldots, m_l$ be n-bit strings of Hamming weight δN. Let C be the direct product code. Then, for every $i \in [l]$, we have*

$$\mathbf{Pr}_{s \in \{0,1\}^{nk}}[C(m_i)[s] = 0^k] = (1-\delta)^k \geq \epsilon.$$

Proof. For a given message m_i, $C(m_i)[s] = 0^k$ when s has an no intersection with the subset of positions in m_i which are 1. The probability of this event is exactly $(1-\delta)^k$ which is at least ϵ. □

Lemma II.C.5. *Let $M = \{m_1, \ldots, m_l\}$ be the set of all N-bit strings of Hamming weight δN, where $\delta \leq 1/17$. Then there is a subset $N \subseteq M$ of size at least $2^{\delta N-1}$ such that any two messages in N have Hamming distance at least δN.*

Proof. Consider a graph with the messages in M denoting the vertices and there is an edge between two messages if they differ in less than δN positions. Set I to be an independent set of this graph, chosen as follows: pick a vertex v and place it in I, delete v and its neighboring vertices; repeat until no vertices are left.

The size of I is lowerbounded by $|M|/(\Delta + 1)$, where Δ is the maximum degree of the graph. We have

$$\begin{aligned}
\Delta &= \sum_{i=1}^{\delta N/2} \binom{\delta N}{i}\binom{(1-\delta)N}{i} \\
&\le \binom{(1-\delta)N}{\delta N/2} \sum_{i=1}^{\delta N/2} \binom{\delta N}{i} \quad \text{(since } \delta \le 1/2\text{)} \\
&\le \binom{(1-\delta)N}{\delta N/2} \cdot 2^{\delta N} \\
&\le 2^{\delta N} \cdot \binom{N}{\delta N/2}.
\end{aligned}$$

This yields

$$\begin{aligned}
|N| &\ge \frac{\binom{N}{\delta N}}{\Delta + 1} \\
&\ge \frac{\binom{N}{\delta N}}{2 \cdot 2^{\delta N} \cdot \binom{N}{\delta N/2}} \\
&= \frac{1}{2^{\delta N+1}} \cdot \frac{N(N-1)\dots(N-\delta N+1)}{N(N-1)\dots(N-\delta N/2+1)} \cdot \frac{(\delta N/2)!}{(\delta N)!} \\
&= \frac{1}{2^{\delta N+1}} \cdot \frac{(N-\delta N/2)(N-\delta N/2-1)\dots(N-\delta N+1)}{(\delta N)(\delta N-1)\dots(\delta N/2+1)} \\
&\ge \frac{(1/\delta - 1)^{\delta N/2}}{2^{\delta N+1}} \\
&\ge 2^{\delta N-1} \quad \text{(for } \delta \le 1/17\text{)}.
\end{aligned}$$

□

Proof of Theorem II.C.3. The proof is immediate from Lemma II.C.4 and Lemma II.C.5. □

The above theorem showed that for a fixed $\delta \le 1/17$, if ϵ is very small ($\delta < \frac{\log 1/\epsilon}{2k}$), then there could be exponentially many messages such that all these messages agrees with the given word on at least ϵ fraction of the inputs and moreover any pair of messages disagree on at least δ fraction of the positions. This implies that for these parameters the direct product code can only be approximately list decoded with extremely large list size. Having established this fact, let us only concentrate on the case when ϵ is large, say $\delta \ge \frac{2\log 1/\epsilon}{k}$. Note that there is a gap in the analysis which is unaccounted for. We hope to address this gap in the future.

The following theorems establish a combinatorial bound on the approximate, local, list-decodability of direct product codes.

Theorem II.C.6 (List size for direct product code (lower bound)). *Let $n, k \in \mathbb{N}$, $\delta \in (0,1]$, $N = 2^n$, $\Sigma = \{0,1\}^k$, and $C : \{0,1\}^N \rightarrow \Sigma^{N^k}$ be the k-wise direct product code. Let $\epsilon > 2^{-N/4}$ and $\delta < 1/2$. There exists a string $s \in \Sigma^{N^k}$ and $t \geq 1/\epsilon$ messages $m_1, ..., m_t$ such that*

1. *$\forall i \in \{1, ..., t\}, \Delta(C(m_i), s) < (1-\epsilon) \cdot N^k$, and*
2. *$\forall i, j \in \{1, ..., t\}, i \neq j, \Delta(m_i, m_j) > \delta N$,*

Proof. Let us arbitrarily partition the set of the k-tuples $(x_1, ..., x_k) \in \{0,1\}^{nk}$ that is used to index into any codeword, into $t = 1/\epsilon$ subsets $S_1, ..., S_t$ each containing at least ϵ fraction of the k-tuples. Given that $\epsilon > 2^{-N/4}$, there exists t messages $m_1, ..., m_t$ such that for any pair of distinct message m_i, m_j, $\Delta(m_i, m_j) > \delta \cdot N$. Given this, we can construct s in the following manner:

$$\forall (x_1, ..., x_k) \in S_i, \quad s[(x_1, ..., x_k)] = m_i[x_1].m_i[x_2]...m_i[x_k]$$

This is sufficient to ensure property (1) of the theorem. This concludes the proof. $\square$

Theorem II.C.7 (List size for direct product code (upper bound)). *Let $n, k \in \mathbb{N}$, $\delta \in (0,1]$, $N = 2^n$, $\Sigma = \{0,1\}^k$, and $C : \{0,1\}^N \rightarrow \Sigma^{N^k}$ be the k-wise direct product code. Let $\delta \geq \frac{2\log 2/\epsilon}{k}$. For any string $s \in \Sigma^{N^k}$ and messages $m_1, ..., m_t$ such that*

1. *$\forall i \in \{1, ..., t\}, \Delta(C(m_i), s) < (1-\epsilon) \cdot N^k$, and*
2. *$\forall i, j \in \{1, ..., t\}, i \neq j, \Delta(m_i, m_j) > \delta N$,*

then $t \leq 2/\epsilon$.

Proof. For the sake of contradiction, assume that for any string $s \in \Sigma^{N^k}$ there are t messages $m_1, ..., m_t$ messages with $t > 2/\epsilon$ such that these messages satisfy properties (1) and (2). Fix the string s and any message m let A_m denote the subset of k-tuples $(x_1, ..., x_k) \in \{0,1\}^{nk}$ such that $s[(x_1, ..., x_k)] = m[x_1]...m[x_k]$. Note that $\forall i \in \{1, ..., t\}, |A_{m_i}|/2^{nk} \geq \epsilon$. We will need the following lemma for further discussion.

Lemma II.C.8. *For any pair of messages $m_i, m_j \in \{m_1, ...m_t\}, i \neq j$,*

$$|A_{m_i} \cap A_{m_j}|/2^{nk} < \epsilon^2/4$$

Proof. Given any pair of messages m_1, m_2 such that $\Delta(m_1, m_2) > \delta N$, their codewords can agree on at most $(1-\delta)^k < \epsilon^2/4$ fraction of positions. $\square$

Fix any $2/\epsilon \geq i$. From the above claim, we know that for all $j \leq i$, $|A_{m_i} \cap A_{m_j}|/2^{nk} < \epsilon^2/4$. This implies that

$$\frac{|A_{m_i} \cap (A_{m_1} \cup A_{m_2} \cup ... \cup A_{m_{i-1}})|}{2^{nk}} < (\epsilon^2/4) \cdot (2/\epsilon) < \epsilon/2$$

On the other hand, we know that $|A_{m_i}|/2^{nk} \geq \epsilon$. This implies that each A_{m_i} contains at least $\epsilon/2$ fraction of k-tuples which are not contained in $A_{m_1} \cup ... \cup A_{m_{i-1}}$. This implies that

$$\frac{|A_{m_1} \cup ... \cup A_{m_{2/\epsilon}}|}{2^{nk}} > 1$$

which gives us a contradiction. $\square$

Suppose the approximate decoding requires to produce a message which agrees with the original message on at least $(1-\delta)$ fraction of the positions when given a corrupted codeword which is corrupted in at most $(1-\epsilon)$ fraction of the positions. Then the above theorem essentially says that it is only possible to output a list of $\Theta(1/\epsilon)$ messages such that at least one of them satisfies the criterion, where ϵ and δ are related as $\delta \geq \frac{2\log(2/\epsilon)}{k}$.

II.D Uniform Direct Product Theorems

Having established connections with direct product codes, let us revisit the question why we failed to obtain a strong direct product theorems in the uniform setting. In order to prove a direct product theorem with respect to computing functions $f : \{0,1\}^n \rightarrow \{0,1\}$ within some computational model, we assume that there is a machine which solves the direct product function f^k on some ϵ fraction of the inputs and then construct a machine which computes the function f on at least $(1-\delta)$ fraction of inputs. Here ϵ and δ are related as $\delta = \Theta\left(\frac{\log 1/\epsilon}{k}\right)$. In the previous section we saw that for these parameters the analogous direct product code is not uniquely decodable but only list decodable with list size $\Theta(1/\epsilon)$. This translates to the fact that given a machine which computes f^k on at least ϵ fraction of the inputs, then there is a list of $\Theta(1/\epsilon)$ machines such that at least one of them computes f on at least $(1-\delta)$ fraction of the inputs. Note that a string of length $\Theta(\log 1/\epsilon)$ can be used to point out the correct machine in the list. With this observation, we can state the previous statement as: given a machine which computes f^k on at least ϵ fraction of the inputs, then there is machine and a string of length $\Theta(\log 1/\epsilon)$ such that this machine when given this string computes f on at least $(1-\delta)$ fraction of the inputs. This then allows us to generalize the previous statement to general functions instead of functions with fixed input length. Here is what we get for general function: Given a function $f : \{0,1\}^* \rightarrow \{0,1\}$, for all sufficiently large n, if there is a machine which computes f^k on at least ϵ fraction of inputs of size nk, then there is a machine and a string of length $\Theta(\log 1/\epsilon)$ such that this machine when given this string computes f on at least δ fraction of inputs of size n. This string for the machine computing f can be interpreted as advice for the machine. Note that this advice string is the same for any input of length n. Interestingly, this is precisely the statement of the direct product theorem in the nonuniform model of computation which essentially says that if a function is hard to compute for Turing machines with advice which is the same for all inputs of the same size then the direct product function is even harder for Turing machines with/without advice. As the size of the advice of Turing machines grows smaller, the bound on the provable hardness of the direct product function decreases. Finally, with no advice which is precisely the uniform setting, we were able to construct a function which was hard with respect to Turing machines but its direct product can be shown to be easy.

In the nonuniform setting, let us informally refer to the length of advice allowed to a Turing machine as the "nonuniformity". We have shown that a strong direct product theorem is not possible in the uniform setting. We also showed that for the parameters of interest, that is $\delta = \Theta\left(\frac{\log 1/\epsilon}{k}\right)$, the amount of nonuniformity needed is $\Omega(\log(1/\epsilon))$. Abusing notation, we let "unifrom direct product theorem" to mean a direct product theorem where the nonuniformity is the least possible. More specifically, here is the formal statement for the uniform direct product theorem. We will prove this theorem in the next Chapter.

Theorem II.D.1 (Uniform Direct Product Theorem). *Let $f : \{0,1\}^* \to \{0,1\}$ be a function, $\epsilon, \delta : \mathbb{N} \to [0,1]$, $k \in \mathbb{N}$. For all sufficiently large n, if f is $\delta(n)$-hard for Turing machines with $\Theta(\log(1/\epsilon(n)))$ advice and running time $t(n)$, then f^k is $(1-\epsilon(n))$-hard for Turing machines with running time $t(n) \cdot poly(\epsilon(n), \delta(n), 1/n, 1/k)$.*

II.E XOR Codes

In the previous sections, we studied direct product codes and its connection with direct product theorems. This was useful in motivating and formalizing uniform direct product theorems which will be proved in the next Chapter. In this section we look at the analogous XOR code and its connection with the XOR Lemma. Here we will formalize the analogous uniform XOR lemma which has interesting applications in Average-case complexity for showing hardness amplification.

The XOR code is defined similarly to the direct product code. Here is the formal description.

Definition II.E.1 (XOR Code)**.** Let $n, k \in \mathbb{N}$, and $N = 2^n$. The k-XOR Code $C : \{0,1\}^N \to \{0,1\}^{N^k}$ is defined as follows: Given a message $m \in \{0,1\}^N$ in binary, the encoding of the message $C(m)$ is also binary. Let the message be indexed by strings in $\{0,1\}^n$, where the bit indexed by $x \in \{0,1\}^n$ is denoted by $m[x]$. The codeword is indexed by a k-tuple of strings in $\{0,1\}^n$. An alphabet in the codeword indexed by $(x_1, ..., x_k) \in \{0,1\}^{nk}$ is denoted by $C(m)[(x_1, ..., x_k)]$ and is defined as

$$C(m)[(x_1, ..., x_k)] = m[x_1] \oplus m[x_2] \oplus ... \oplus m[x_k]$$

As for direct product codes, we would be interested in locally, approximate, list decoding of the k-XOR code defined above. In the next subsection we study some list decoding properties of the XOR code.

II.E.1 List-decoding XOR Codes

As in the previous section, we will show that for a fixed value of $\delta \leq 1/17$, if ϵ is small ($\delta < \frac{\log 1/\epsilon}{4k}$), then there could be exponentially many messages such that their codewords agrees with the given word on at least $1/2 + \epsilon$ fraction of positions and every pair of messages disagree on at least δ fraction of positions.

Theorem II.E.2. *Let $\delta \leq \min\{(\ln 1/\epsilon)/(4k), 1/17\}$. There exists a set of at least $2^{\delta N - 1}$ of N-bit messages of pairwise Hamming distance at least δN and such that the XOR encoding of each message has agreement at least $1/2 + \epsilon$ with a string $s \in \Sigma^{N^k}$.*

The proof of this theorem will follow from the two lemmas below.

Lemma II.E.3. *Let $\delta \leq \min\{(\ln 1/\epsilon)/(4k), 1/3\}$ and let $m_1, \ldots, m_l$ be n-bit strings of Hamming weight δN. Let C be the direct product code. Then, for every $i \in [l]$, we have*

$$\mathbf{Pr}_{s \in \{0,1\}^{nk}}[C(m_i)[s] = 0] = 1/2 + (1 - 2\delta)^k/2 \geq 1/2 + \epsilon.$$

Proof. For a given message m_i, $C(m_i)[s] = 0$ when s has an even intersection with the subset of positions in m_i which are 1. The probability of this event is exactly $\sum_{\text{even } i\in[k]} \binom{k}{i}\delta^i(1-\delta)^{k-i}$ which is $1/2 + (1-2\delta)^k/2 \geq 1/2+\epsilon$ when $\delta \leq \min\{(\ln 1/\epsilon)/(4k), 1/3\}$. □

Proof of Theorem II.E.2. The proof is immediate from Lemma II.E.3 and Lemma II.C.5. □

We now show that XOR-codes are approximately list-decodable for an appropriate choice of parameters.

Theorem II.E.4 (List size for XOR code (upper bound))**.** *Let $n, k \in \mathbb{N}$, $\delta \in (0,1]$, $N = 2^n$, $\Sigma = \{0,1\}^k$, and $C : \{0,1\}^N \to \{0,1\}^{N^k}$ be the k-wise XOR code. Let $\delta \geq \frac{\log 1/\epsilon}{k}$. For any string $s \in \{0,1\}^{N^k}$ and messages $m_1, ..., m_t$ such that*

1. *$\forall i \in \{1,...,t\}, \Delta(C(m_i), s) < (1/2-\epsilon)\cdot N^k$, and*
2. *$\forall i,j \in \{1,...,t\}, i \neq j, \Delta(m_i, m_j) > \delta N$,*

then $t \leq 1/(3\epsilon^2)$.

Proof. For every $w \in \{0,1\}^{nk}$, let

$$\epsilon_w = \mathbf{Pr}_{i\in[t]}[C(m_i)[w] = s[w]] - \mathbf{Pr}_{i\in[t]}[C(m_i)[w] \neq s[w]] = \frac{1}{t}\sum_{i\in[t]}(-1)^{C(m_i)[w]\oplus s[w]}.$$

Observe that $\mathbf{Exp}_{w\in\{0,1\}^{nk}}[\epsilon_w] \geq 2\epsilon$. So, we get that $4\epsilon^2 \leq (\mathbf{Exp}_w[\epsilon_w])^2$. By Jensen's inequality, the latter is at most $\mathbf{Exp}_w[(\epsilon_w)^2]$.

We have

$$\begin{aligned}
\mathbf{Exp}_w[(\epsilon_w)^2] &= \mathbf{Exp}_w\left[\frac{1}{t^2}\sum_{i,j}(-1)^{C(m_i)[w]\oplus C(m_j)[w]}\right] \\
&= \mathbf{Exp}_w\left[\frac{1}{t^2}\sum_{i,j}(-1)^{C(m_i\oplus m_j)[w]}\right] \\
&= \frac{1}{t^2}\cdot \mathbf{Exp}_w\left[\sum_i(-1)^0 + \sum_{i\neq j}(-1)^{C(m_i\oplus m_j)[w]}\right] \\
&= \frac{1}{t} + \frac{1}{t^2}\cdot\sum_{i\neq j}\mathbf{Exp}_w\left[(-1)^{C(m_i\oplus m_j)[w]}\right].
\end{aligned}$$

Next we bound the quantity $\mathbf{Exp}_w\left[(-1)^{C(m_i \oplus m_j)[w]}\right]$ in the expression above.

Claim II.E.5. *For any $i \neq j$, we have* $\mathbf{Exp}_w\left[(-1)^{C(m_i \oplus m_j)[w]}\right] \leq (1-2\delta)^k$.

Proof. First, observe that by the assumption on pairwise distance between messages, we have that the string $m' = m_i \oplus m_j$ has relative Hamming weight $\rho \geq \delta$. We need to compute the probability that a random k-tuple $w \in \{0,1\}^{nk}$ hits an even number of 1s in the string m' minus the probability that it hits an odd number of 1s. This is exactly

$$\begin{aligned}
&\left(\sum_{\text{even } i \in [k]} \binom{k}{i} w^i (1-w)^{k-i} - \sum_{\text{odd } i \in [k]} \binom{k}{i} w^i (1-w)^{k-i}\right) \\
= \ & \sum_{i \in [k]} \binom{k}{i} (-w)^i (1-w)^{k-i} \\
= \ & (1-2w)^k.
\end{aligned}$$

The latter is at most $(1-2\delta)^k$. □

Using the bound from Claim II.E.5, we have

$$\mathbf{Exp}_w(\epsilon_w)^2 \leq \frac{1}{t} + \frac{t(t-1)}{t^2}(1-2\delta)^k \leq \frac{1}{t} + e^{-2\delta k}.$$

Recalling that $\mathbf{Exp}_w(\epsilon_w)^2 \geq 4\epsilon^2$, we obtain

$$t \leq \frac{1}{4\epsilon^2 - e^{-2\delta k}} \leq \frac{1}{3\epsilon^2},$$

as required. □

Next, we obtain a lower bound on the list size for XOR code. We do that by first obtaining a lower bound for a code which is very similar to the XOR code. We call it the Subset XOR code. We then extend this lower bound to the XOR code.

Definition II.E.6 (Subset XOR Code)**.** Let $n, k \in \mathbb{N}$, and $N = 2^n$. The Subset k-XOR Code $C : \{0,1\}^N \to \{0,1\}^{N^k}$ is defined as follows: Given a message

$m \in \{0,1\}^N$ in binary, the encoding of the message $C(m)$ is also binary. Let the message be indexed by strings in $\{0,1\}^n$, where the bit indexed by $x \in \{0,1\}^n$ is denoted by $m[x]$. The codeword is indexed by a subset of strings in $\{0,1\}^n$ of cardinality k. An alphabet in the codeword indexed by $(x_1, ..., x_k) \in \{0,1\}^{nk}$ is denoted by $C(m)[(x_1, ..., x_k)]$ and is defined as

$$C(m)[(x_1, ..., x_k)] = m[x_1] \oplus m[x_2] \oplus ... \oplus m[x_k]$$

Theorem II.E.7 (List size for Subset XOR code (lower bound))**.** *Let $n, k \in \mathbb{N}$, $\delta \in (0,1]$, $N = 2^n$, $\Sigma = \{0,1\}^k$, and $C : \{0,1\}^N \to \{0,1\}^{N^k}$ be the Subset k-wise XOR code. Let $\epsilon > \max\{\binom{N}{k}^{-1/256}, 2^{-N/256}\}$ and $\delta < 1/4$. For any string $s \in \{0,1\}^{N^k}$, there are $t = \Omega(1/\epsilon^2)$ messages $m_1, ..., m_t$ such that*

1. *$\forall i \in \{1, ..., t\}, \Delta(C(m_i), s) < (1/2 - \epsilon) \cdot N^k$, and*
2. *$\forall i, j \in \{1, ..., t\}, i \neq j, \Delta(m_i, m_j) > \delta N$,*

Proof. Let $t = 2T + 1$ be an odd positive integer to be specified later. Pick strings $a_1, \ldots, a_t \in \{0,1\}^N$ uniformly at random. The probability that any two of them are within Hamming distance less than $N/4$ is at most $O(t^2 e^{-(1/4)^2 N/2})$. This probability is less than $o(1/\sqrt{t})$ for $t < e^{N/90}$.

For a given t-tuple $a = (a_1, \ldots, a_t)$, define the function $B_a : \{0,1\}^N \to \{0,1\}$ as follows: for every $r \in \{0,1\}^N$,

$$B_a(r) = Maj_{1 \le i \le t} \langle a_i, r \rangle.$$

We will show that there exists a such that the function B_a evaluated at all strings r of Hamming weight exactly k agrees in at least $1/2 + \Omega(1/\sqrt{t})$ fraction of places with $C(a_i)$, for at least $\Omega(t)$ of a_i's. This will imply the lemma for $\epsilon = \Omega(1/\sqrt{t})$.

The intuition is as follows. Fix a string r of Hamming weight k. Fix an index $i \in [t]$. Randomly choose all the other a_j for $j \neq i$. For each $j \neq i$, the random variable $\langle a_j, r \rangle$ is a fair coin flip (for a random a_j). For different j's, the corresponding random variables are independent. If we flip $2T$ independent fair coins, we get exactly T heads with probability at least $\Omega(1/\sqrt{T})$. Conditioned on getting exactly T heads, we have that $B_a(r) = \langle a_i, r\rangle$, i.e., $B_a(r) = C(a_i)_r$ [3]. By averaging, we can argue that if we randomly fix a_j's for $j \neq i$, then (with probability at least $\Omega(1/\sqrt{T})$) we will have at least $\Omega(1/\sqrt{T})$ fraction of strings r (of Hamming weight k) for which we have $B_a(r) = C(a_i)_r$. For every remaining r, the value $B_a(r)$ is fixed (because of the fixed a_j's for $j \neq i$), but is independent of a_i. The random variables $\langle a_i, r\rangle$ over these remaining r's are uniformly distributed and *pairwise independent*. Hence, a random choice of a_i is likely to result in about $1/2$ of these random variables being equal to the fixed value $B_a(r)$ (and by the Chebyshev inequality, we can bound the probability that this value deviates from the expectation). Thus, for random $a_1, \ldots, a_t$, we are likely to get B_a and $C(a_i)$ agree in about $1/2 + \Omega(1/\sqrt{T})$ fraction of positions. With some extra work, we will show that this happens simultaneously for $\Omega(t)$ different indices $i \in [t]$, which yields many codewords with good agreement with the string B_a. We give a formal argument next.

Given a t-tuple a, we say that a string r is *balanced for a* if the number of indices i with $\langle a_i, r\rangle = 0$ is either T or $T+1$.

Claim II.E.8. *There is a constant c such that for at least $c/\sqrt{t}$ fraction of random t-tuples a, there are at least $c/\sqrt{t}$ fraction of strings $r \in \{0,1\}^N$ of Hamming weight k such that each r is balanced for a.*

Proof. Each fixed nonzero r (of Hamming weight k) is balanced for a with probability at least $c'/\sqrt{t}$ over the choice of a random t-tuple a, for some constant c'. Hence, we have

$$\mathbf{Pr}_{a,r}[r \text{ is balanced for } a] \geq c'/\sqrt{t},$$

where r is a random N-bit string of Hamming weight k. By averaging, the claim follows for $c = c'/2$. □

[3]for a string $r \in \{0,1\}^N$ of hamming weight exactly k, let $x_1, ..., x_k\ in\{0,1\}^n$ denote the indices such that $\forall i, r[x_i] = 1$, then $C(a_i)_r$ denotes $C(a_i)[(x_1, ..., x_k)]$.

For any $i \in [t]$, and any $(t-1)$-tuple $a^{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_t)$ of n-bit strings, we say that a string $r \in \{0,1\}^n$ is *i-balanced for a^{-i}* if the number of $j \in [t] \setminus \{i\}$ with $\langle m_j, r \rangle = 0$ is T.

Claim II.E.9. *Let d be any constant. Suppose $i \in [t]$ and $a^{-i} = (a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_t)$ are such that $d/\sqrt{t}$ fraction of strings $r \in \{0,1\}^N$ of Hamming weight k are i-balanced for a^{-i}. Then for a random $a_i \in \{0,1\}^N$ and the t-tuple $a = (a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_t)$, we have*

$$\mathbf{Pr}_{a_i}\left[B_a(r) = \langle a_i, r \rangle \text{ for at least } \frac{1}{2} + \frac{d/4}{\sqrt{t}} \text{ of } r\text{'s}\right] > 1 - O\left(\frac{t}{m}\right),$$

where $m = \binom{N}{k}$.

Proof. For r's that are i-balanced for a^{-i}, we have $B_a(r) = \langle a_i, r \rangle$ whatever a_i is. Let R be the set of the remaining r's of weight k that are not i-balanced. For each $r \in R$, $B_a(r)$ is determined, and so is independent of the choice of a_i.

Define random variables X_r where $X_r = 1$ if $< a_i, r >= B_a(r)$, and $X_r = 0$ otherwise. We get that, for $r \in R$, these random variables X_r are uniformly distributed and pairwise independent, for a random choice of a_i. We expect $X_r = 1$ for half of the r's. By Chebyshev, for any constant d', the probability that there are fewer than $1/2 - d'/\sqrt{t}$ fraction of $r \in R$ with $X_r = 1$ is less than $O(t/m)$.

Finally, observe that if the latter event does not happen for some a_i, then we have that the agreement between $\langle a_i, r \rangle$ and $B_a(r)$ is at least

$$\frac{d}{\sqrt{t}} + \left(1 - \frac{d}{\sqrt{t}}\right)\left(\frac{1}{2} - \frac{d'}{\sqrt{t}}\right) \geq \frac{1}{2} + \frac{(d/2) - d'}{\sqrt{t}}.$$

The latter can be made at least $1/2 + (d/4)/\sqrt{t}$ by choosing $d' = d/4$. □

By Claim II.E.8, there are at least $c/\sqrt{t}$ fraction of a's with $c/\sqrt{t}$ fraction of strings r balanced for a. Fix any such a. Observe that each r that is balanced for a is also i-balanced for $T + 1 > t/2$ values of i. Since we have $c/\sqrt{t}$ fraction of r's balanced for a, we get by a simple averaging argument that there are at least $t/4$ values of i such that each has at least $(c/4)/\sqrt{t}$ fraction of i-balanced r's. Note that this happens with probability at least $c/\sqrt{t}$ over a's.

On the other hand, Claim II.E.9 implies that the probability (over t-tuples a) that there is at least one i with $(c/4)/\sqrt{t}$ fraction of i-balanced r's, such that for this i, the agreement between $\langle a_i, r\rangle$ and $B_a(r)$ is less than $1/2 + (c/16)/\sqrt{t}$ is at most $O(t^2/m)$. For $t = o(m^{1/3})$, this probability is less than $o(1/\sqrt{t})$.

Thus there is at least $c/\sqrt{t} - o(1/\sqrt{t}) \geq (c/2)/\sqrt{t}$ fraction of a's such that, for each of these a's, we have that

1. there are at least $t/4$ values of i, each having at least $(c/4)/\sqrt{t}$ fraction of i-balanced r's, and

2. for every such i, the agreement between $\langle a_i, r\rangle$ and $B_a(r)$ is at least $1/2 + (c/16)/\sqrt{t}$.

Recall that for all but $o(1/\sqrt{t})$ fraction of $a = (a_1, \ldots, a_t)$ we have that the pairwise Hamming distance between a_i and a_j is at least δN, for all $i \neq j$ in $[t]$. Therefore, there must exist a choice of $a = (a_1, \ldots, a_t)$ and a subset I of $t/4$ of the i's such that, for each $i \in I$, the agreement between $code(a_i)$ and B_a is at least $1/2 + (c/16)/\sqrt{t}$, and the pairwise Hamming distance between any a_i and a_j, for $i \neq j$, is at least δn. Setting $t = ((c/16)/\epsilon)^2$ concludes the proof. $\square$

Now we reduce the case of XOR codes to the case of Subset XOR codes, obtaining the following.

Theorem II.E.10. *Let* $m = \binom{N}{k}$, $\epsilon > \max\{m^{-1/256}, 2^{-N/256}\}$, $k^2/N \leq o(\epsilon)$, *and* $\delta < 1/4$. *Let* $C : \{0,1\}^N \to \{0,1\}^{N^k}$ *be the XOR-code. Then there exists a string* $B \in \{0,1\}^{N^k}$ *and* $t = \Omega(1/\epsilon^2)$ *messages* $m_1, \ldots, m_t$ *of pairwise Hamming distance at least* δN *such that the agreement between* B *and* $C(m_i)$ *is at least* $1/2 + \Omega(\epsilon)$ *for each* $i \in [s]$.

Proof. Recall that the XOR encoding of an N-bit message m is the sequence of $m[x_1] \oplus \cdots \oplus m[x_k]$ over all k-tuple $(x_1, \ldots, x_k) \in \{0,1\}^{nk}$. The fraction of those k-tuples $(x_1, \ldots, x_k)$ that contain some index $x_j \in \{0,1\}^n$ more than once is at most k^2/N, which is $o(\epsilon)$ by our assumption.

Ignoring the k-tuples with repeats, we can partition the remaining k-tuples into ℓ blocks where each block contains $\binom{N}{k}$ tuples corresponding to distinct k-size subsets of $\{0,1\}^n$. For each such block, the XOR encoding of a given message m (restricted to the k-tuples in the block) coincides with the Subset XOR encoding of m. So, the XOR encoding of m restricted to the k-tuples with Subset elements is just a concatenation of ℓ copies of the Subset XOR encoding of m.

By Theorem II.E.7, there is a collection of $\Omega(1/\epsilon^2)$ N-bit messages (pairwise Hamming distance δN apart) and a string B' such that the Subset XOR encoding of each message agrees with B' in at least $1/2 + \Omega(\epsilon)$ fraction of positions. Let B'' be the string obtained as a concatenation of ℓ copies of the string B'. It follows that for the same collection of messages, their XOR encodings will agree with the string B'' in at least $1/2 + \Omega(\epsilon)$ fraction of positions, when the positions are restricted to the k-tuples with Subset elements. Let us now pad B'' with enough 0's to get the string of length N^k. Let us call the new string B. We have that the XOR encodings of our messages will agree with B in at least $1/2 + \Omega(\epsilon) - k^2/N \geq 1/2 + \Omega(\epsilon)$ fraction of positions. $\square$

II.F Summary of List-decoding Bounds

We showed that if δ smaller than $\min\left(\frac{1}{17}, \frac{\log(1/\epsilon)}{4k}\right)$, then the list size for approximate list decoding both direct product and XOR code is exponentially large in the message size. On the other hand, when δ is larger than $\frac{2 \cdot \log(1/\epsilon)}{k}$, then the list size can be shown to be $\Theta(1/\epsilon)$ for the direct product code and $\Theta(1/\epsilon^2)$ for the XOR Code.

Acknowledgements : This Chapter is, in part, based on the paper “Approximately List-decoding Direct Product Codes and Uniform Hardness Amplication” co-authored with Russell Impagliazzo and Valentine Kabanets. In Proceedings of the Forty-Seventh Annual IEEE Symposium on Foundations of Computer Science (FOCS’06), pages 187–196, 2006.

III

Uniform Direct Product Theorems

(This Chapter is a joint work with Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson.)

III.A Introduction

Applications such as cryptography and derandomization require reliably hard problems, ones that cannot be solved by any fast algorithm with even a non-trivial advantage over random guessing. Direct-product theorems are a primary tool in hardness amplification, allowing one to convert problems that are somewhat hard into problems that are more reliably hard. In a direct-product theorem, we start with a function f such that any feasible algorithm has a non-negligible chance of failing to compute $f(x)$ given a random x. We then show that no feasible algorithm can, given multiple instances of the problem $x_1, \ldots, x_k$, compute all of the values $f(x_i)$, with even a small probability of success. (Usually, the x_i's are chosen independently, but there are also derandomized direct-product theorems where the x_i's are chosen pseudo-randomly.) Many strong direct product theorems are known for non-uniform models, such as Boolean circuits [Yao82, Lev87, GNW95, Imp95b, IW97, STV01]. Unfortunately, in general, direct-product theorems fail in completely uniform models such as probabilistic computation.

However, Trevisan [Tre05] pointed out that proofs of direct product theorems correspond to (approximate) error-correction of sparse codes. Using this view, we think of a function f as being encoded by $Code(f) = f^k$, its values on all k-tuples. We seek a decoding algorithm which will generate efficient circuit(s) for f (on most inputs), given access to a circuit C' which is a highly corrupted codeword, agreeing with f^k only on an ϵ-fraction of all k-tuples. (Note that this code, like any code that can be computed locally with oracle access to f, has extremely poor distance. This precludes exact decoding, i.e., recovering a circuit that computes f on all inputs, but not approximate decoding.)

The strictly uniform direct-product theorem fails because these codes are not uniquely decodable. A circuit C' might agree on ϵ-fraction of k-tuples for each of $1/\epsilon$ different functions. Thus list decoding is essential, and one can quantify uniformity in terms of the list size. However, the non-uniform direct-product theorems yield list sizes which are all *exponential* in $1/\epsilon$. In contrast, a strong uniform direct-product theorems should have the list size which is polynomial in $1/\epsilon$. [IJK06] gave the first such proof of the direct-product theorem. However, their reduction was quite complex and fell short of the information-theoretic bounds in many respects.

Here, we give a new uniform direct-product theorem that has the following features:

- **Optimality:** The parameters achieved by our list decoding algorithm are information theoretically optimal (to within constant factors).
- **Efficiency:** The decoding algorithm is simply a projection, namely implementable in uniform NC^0 with oracle access to the corrupted circuit C'. The circuits it produces are implementable in uniform AC^0. Thus, our hardness amplification applies to much simpler uniform classes than P.

- **Simplicity:** Both the decoding algorithm and the proof of correctness are extremely simple (even when compared with proofs in the non-uniform setting!).

- **Generality:** The decoding algorithm and its proof turns out to work without change for a general family of codes of which the above direct-product code is just an example. We define this class of *intersection codes*, which is simply specified by the family of k-subsets used to record values of f in $Code(f)$. We explain how the quality of the decoding (and thus of the amplification) depend on the sampling properties of the family of sets, and of their pairwise intersections.

- **Derandomization:** As an immediate bonus of the above setting we get the first derandomized direct-product theorems in the uniform setting. A direct application of the above intersection codes to subspaces yields amplification with input size $O(n)$, instead of the trivial bound of $O(kn)$ when using all subsets. In a more sophisticated application, using a concatenation of two intersection codes, we get similar savings in randomness, but with hardly any loss in other parameters.

- **Consequences:** As observed by [TV02, Tre05], efficient list-decoding has the same consequences as unique decoding in terms of hardness amplification within many natural complexity classes, e.g., $\mathsf{NP}, \mathsf{P}^{\mathsf{NP}\|}, \#\mathsf{P}, \mathsf{PSPACE}$ and EXP.

III.A.1 Statement of the Uniform Direct-Product theorem

We say that a circuit C ϵ-computes a function F if $C(z) = F(z)$ for at least ϵ fraction of inputs z. A function F is $(1 - \epsilon)$-hard for size $t(n)$ if no circuit of size $t(n)$ ϵ-computes F.

Following [TV02], we define the "semi-uniform" class $\mathsf{BPP}//\log$ as the class of probabilistic algorithms with advice of length $O(\log n)$ that depends on the random coin tosses of the algorithm, but not on the input. We can view such an algorithm as producing a polynomial-sized list of polynomial-size circuits: the algorithm then is judged by how well the best circuit on its list does. A probabilistic polynomial-time algorithm with advice, $A(x, r, z)$, ϵ-computes F if, for every length n, there is a function $z(r)$ taking a polynomial-size string r to a logarithmic length output, so that $\mathbf{Pr}_{x,r}[A(x, r, z(r)) = F(x)] \geq \epsilon$. A function F is $(1 - \epsilon)$-hard for $\mathsf{BPP}//\log$ if no such algorithm and function $z(r)$ exist. For superpolynomial time complexity $t = t(n)$, we can generalize in the obvious way to the class $\mathsf{BPTIME}(\mathrm{poly}(t))//\log t$.

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the k-wise direct-product function f^k maps every k-set $(x_1, \ldots, x_k)$ of n-bit strings (ordered according to some fixed ordering of the universe $\{0,1\}^n$) to the k-tuple $(f(x_1), \ldots, f(x_k))$.[1] One of our main results is the following.

Theorem III.A.1 (Uniform Direct-Product Theorem)**.** *There is an absolute constant $c > 0$ so that for any functions $\delta = \delta(n)$, $k = k(n)$, $t = t(n)$, and $\epsilon = \epsilon(n) \geq e^{-\delta k/c}$ and $\epsilon > t^{-1/c}$, if f is δ-hard for $\mathsf{BPTIME}(poly(t(nk)))//\log t$, then f^k is $(1 - \epsilon)$-hard for $\mathsf{BPTIME}(poly(t))//\log t$.*

The proof is via the following reconstruction algorithm, which is information-theoretically optimal up to constant factors.

Theorem III.A.2 (Approximate list-decoding algorithm)**.** *There is a constant c and a probabilistic algorithm $\mathcal{A}$ with the following property. Let $k \in \mathbb{N}$, and $0 < \epsilon, \delta < 1$ be such that $\epsilon > e^{-\delta k/c}$. Let C' be a circuit that ϵ-computes the Direct-Product f^k, for some Boolean function $f : \{0,1\}^n \to \{0,1\}$. Given such a circuit C', algorithm $\mathcal{A}$ outputs with probability $\Omega(\epsilon)$ a circuit C that $(1 - \delta)$-computes f. The algorithm $\mathcal{A}$ is a uniform randomized NC^0 algorithm (with one C'-oracle gate), and the produced circuit C is an AC^0 circuit of size $\mathrm{poly}(n, k, \log 1/\delta, 1/\epsilon)$ (with C'-oracle gates).*

[1]The usual definition of k-wise direct product is in terms of k-tuples rather than k-sets, but it is easily seen to be equivalent, by randomly reordering the input tuple.

The circuit output by algorithm $\mathcal{A}$ will have the following structure. Fix $s = k/2$. Let $A = (a_1, \ldots, a_s)$ be an s-subset of $\{0,1\}^n$, and let $v = (v_1, \ldots, v_s)$ be an s-bit string. For intuition, imagine that $v_i = f(a_i)$ for all $1 \leq i \leq s$.

We define the following randomized **circuit $\mathbf{C_{A,v}}$:**

"On input $x \in \{0,1\}^n$, check if $x = a_i$ for some $a_i \in A$; if so, then output v_i. Otherwise, repeatedly sample random k-sets B such that $A \cup \{x\} \subseteq B$, discarding any B where C' is inconsistent with our answers v for A (i.e., where $C'(B)|_A \neq v$). For the first consistent B, output $C'(B)|_x$. Produce some default (error) output if no consistent B is found even after $100 \cdot (\ln 1/\delta)/\epsilon$ iterations."

Intuitively, if each $v_i = f(a_i)$, we are checking consistency on a subset of inputs to see whether we believe $C'(B)$ on another input. In addition to having the correct values, the algorithm $C_{A,v}$ requires that $C'(B)$ is correct for many B with $A \subset B$ (and thus, the algorithm usually finds a consistent B). Finally, we need that consistency implies correctness: for most B for which $C'(B)$ is consistent with v on A, it should be the case that $C'(B)$ is (almost) the same as $f|B$. We show that these three properties are strongly correlated, so that they simultaneously happen with good probability.

The main algorithm **algorithm $\mathcal{A}$** is simply:

"Pick at random a k-set B_0, an s-subset $A \subseteq B_0$. Set $v = C'(B_0)|_A$. Output the circuit $C_{A,v}$."

III.A.2 Generalized direct-product encoding: intersection codes

Our analysis for direct products immediately generalize to decoding the following form of possibly derandomized direct product codes. Let $f : U \to R$, where U is some universe. Usually, U will be $\{0,1\}^n$, or $\mathbb{F}_q^m$, an m-dimensional vector space over a finite field $\mathbb{F}_q$. The range R is an arbitrary set ($R = \{0,1\}$ for Boolean f). Without loss of generality, we identify an s-tuple of elements of U with the s-subset of elements appearing in the tuple.

For $1 \leq s < k \in \mathbb{N}$, a *$k$-intersection code* is specified by two families of subsets of U, $\mathcal{T}$ a family of k-subsets of U, and $\mathcal{S}$, a family of s-subsets of U (with $s < k$). The family $\mathcal{S}$ is only used in the analysis. The encoding of f using $Code = Code(\mathcal{T}, \mathcal{S})$ is the restriction of the direct product f^k to sets $B \in \mathcal{T}$.

Our two running examples of these families are:

- **Independent**: $\mathcal{T}$ are all k-subsets of U, and $\mathcal{S}$ are all s-subsets of U; we only use the case $s = k/2$.
- **Subspaces**: We identify U with the vector space $\mathbb{F}_q^m$. For positive integers $d \geq 8$ and $r = d/2$, we take $\mathcal{T}$ to be all d-dimensional affine subspaces of U, and $\mathcal{S}$ to be all r-dimensional affine subspaces of U. Here we have $k = q^d$ and $s = q^r = \sqrt{k}$.

The Independent example is the k-wise direct-product function considered earlier. The Subspaces example will give us a derandomized version of the direct-product theorem, where inputs of f^k will be all points in a given affine d-dimensional subspace of U. Note that to specify $k = q^d$ such points, we only need to specify the $d + 1$ vectors of U that define the d-dimensional affine subspace (d basis vectors plus a shift vector). In our case, d and r will be constants, and so these affine subspaces are specified with only $O(n)$ bits.

The code $Code$ is δ*-approximately* (ϵ, ℓ)*-list decodable* if for every function $C' : \mathcal{T} \to R^k$ there is a collection of at most ℓ functions $g_1, g_2, \cdots, g_\ell$ such that, for every function $f : U \to R$, if $Code(f)$ ϵ-agrees with C', then f will $(1 - \delta)$-agree with some g_i, for $1 \leq i \leq \ell$. The code $Code$ is *efficiently locally decodable* if there is an efficient algorithm that uses oracle access to C' to generate circuits for the functions g_i's (which also use that oracle).

Our decoding algorithm for $Code(\mathcal{S}, \mathcal{T})$ is exactly the same as the algorithm $\mathcal{A}$ described in the previous section, with sets A coming from $\mathcal{S}$, and sets B from $\mathcal{T}$. We show that this algorithm $\mathcal{A}$ produces a good circuit for f, provided that families $\mathcal{S}, \mathcal{T}$ satisfy certain sampling conditions. In particular, we prove the following.

Theorem III.A.3. *Both Independent and Subspaces codes are efficiently, locally, δ-approximately $(\epsilon, O(1/\epsilon))$-list decodable, where*

- ***Independent:*** $\delta = O((\log 1/\epsilon)/k)$,
- ***Subspaces:*** $\delta = O(1/(\epsilon^2 k^{1/4}))$.

Moreover, the decoder for the Independent code is a uniform randomized NC^0 *algorithm that outputs* AC^0 *circuits.*[2]

Informally, the following properties of $\mathcal{T}, \mathcal{S}$ are all we use:

Computational assumptions: It is efficiently possible to: choose B uniformly in $\mathcal{T}$; given $B \in \mathcal{T}$, uniformly pick $A \in \mathcal{S}$ with $A \subset B$; given $A \in \mathcal{S}$ and $x \in U \setminus A$, uniformly pick $B \in \mathcal{T}$ with $A \cup \{x\} \subset B$.

Symmetry: For a fixed $B \in \mathcal{T}$, for a random $A \in \mathcal{S}$ with $A \subset B$, the elements of A are individually uniform over B. For a fixed $A \in \mathcal{S}$, and random $B \in \mathcal{T}$ with $A \subset B$, the elements in $B \setminus A$ are individually uniform over $U \setminus A$.

Sampling: For a fixed $B \in \mathcal{T}$ and any sufficiently large subset $W \subset B$, with high probability over a random $A \in \mathcal{S}, A \subset B$, $|A \cap W|/|A|$ is approximately the same as $|W|/|B|$. For a fixed $A \in \mathcal{S}$, and any sufficiently large subset $H \subset U \setminus A$, with high probability over a random $B \in \mathcal{T}, A \subset B$, we have that $|(B \setminus A) \cap H|/|B \setminus A|$ is approximately the same as $|H|/|U \setminus A|$.

III.A.3 Concatenated codes and hardness condensing

We also prove a stronger version of Theorem III.A.3 for the case where we allow an oracle circuit C' for the direct-product f^k to be only *approximately* correct on at least ϵ fraction of inputs to f^k. More precisely, we allow a circuit C' such that, for at least ϵ fraction of $T \in \mathcal{T}$, $C'(T)$ and $f^k(T)$ agree on at least $(1 - \delta')$ fraction of elements of T. Note that the usual version of direct-product

[2] *This yields a much simpler construction of non-binary codes, locally list-decodable in uniform randomized* AC^0*, than the one given by [GGH$^+$07].*

decoding assumes $\delta' = 0$. Given such a circuit C', we show how to obtain a circuit C which $(1-\delta)$-computes f, for $\delta = O(\delta')$.

This relaxed notion of approximate list decoding can be formalized as follows. The code *Code* is (δ, δ')*-approximately* (ϵ, ℓ)*-list decodable* if for every function $C' : \mathcal{T} \to R^k$ there is a collection of at most ℓ functions $g_1, g_2, \cdots, g_\ell$ such that, for every function $f : U \to R$, if the k-tuples $f^k(T)$ and $C'(T)$ $(1-\delta')$-agree on at least ϵ fraction of sets $T \in \mathcal{T}$, then f will $(1-\delta)$-agree with some g_i, for $1 \leq i \leq \ell$. *Efficient local decodability* means, as before, that a collection of circuits for such g_i's can be efficiently generated, given oracle access to a circuit C'.

We prove the following "approximate" version of Theorem III.A.3.

Theorem III.A.4. *Both Independent and Subspaces codes are efficiently, locally,* $(\delta, \Omega(\delta))$*-approximately* $(\epsilon, O(1/\epsilon))$*-list decodable, where*

- ***Independent:*** $\delta = O((\log 1/\epsilon)/k)$,
- ***Subspaces:*** $\delta = O(1/(\epsilon^2 k^{1/4}))$.

While interesting in its own right, Theorem III.A.4 will also allow us to obtain a strong derandomized version of uniform direct product theorem for a Boolean function $f : \{0,1\}^n \to \{0,1\}$. The direct product code using affine subspaces already yields a harder function on inputs of size $O(n)$, but only with hardness polynomial in $1/k$. In non-uniform settings, there are derandomized direct product theorems with input size $O(n)$ and hardness exponentially small in n([IW97, STV01]). We will be able to meet this goal partially: we define a function h of hardness $\epsilon = e^{-\Omega(\sqrt{n})}$ with input size $O(n)$ and $k = O(\log 1/\epsilon)$.

The function h combines the two direct product theorems. For $k = \sqrt{n}$ and a field $\mathbb{F}$ of size $q = 2^{\sqrt{n}}$, h is the restriction of f^k to k-subsets of inputs that all lie within a low dimensional affine subspace of $\mathbb{F}^{\sqrt{n}}$. We can specify the input to h by specifying a basis for the subspace with $O(n)$ bits, and then specifying $\sqrt{n}$ elements of the subspace in terms of this basis, using $O(\sqrt{n})$ bits each. We view h as a concatenation of two encodings. First, for $K = q^d = 2^{O(\sqrt{n})}$, we think of the K-direct product code for f using affine subspaces. This code, for each subspace, lists the value of f for all inputs in the subspace. This would be very large, so instead, we encode each block of the subspace code with the Independent k-direct product code, for $k = \sqrt{n}$, listing the values on subsets within each a affine subspace. To decode, we use the Independent direct product decoding within a given affine subspace as a subroutine in the affine subspace decoding procedure. Since the Independent direct product decoding is only approximate, we need Theorem III.A.4 to handle errors created in the decoding of the inner code.

Theorem III.A.5. (Uniform Derandomized Direct Product Theorem) *There is an absolute constant $c > 0$ so that for any constant $0 < \delta < 1$, and any functions $t = t(n)$, $k = k(n)$, $\epsilon = \epsilon(n) \geq \max\{e^{-\delta k/c}, e^{-\Omega(\sqrt{n})}, t^{-1/c}\}$, and $K = K(n) = O(1/(\epsilon\delta)^8)$, if $f : \{0,1\}^n \to \{0,1\}$ is δ-hard for $\mathsf{BPTIME}(t)//\log t$, then the function h defined from f as described above is $(1-\epsilon)$-hard for $\mathsf{BPTIME}(t^{1/c})//(1/c)\log t$. The input size of h is $O(n)$.*

We give an interpretation of Theorem III.A.5 in terms of "hardness condensing" in the spirit of [BOS06]. We obtain some form of "hardness condensing" with respect to $\mathsf{BPTIME}(t)//\log t$. For an affine subspace $B \in \mathcal{T}$, think of $g(B) = f|_B$ as the truth table of the Boolean function mapping $b \in B$ to $f(b)$. Since B is an affine d-dimensional subspace, each element of B can be described by a d-tuple of field elements $(\alpha_1, \ldots, \alpha_d) \in \mathbb{F}_q^d$, and so each $f|_B : \mathbb{F}_q^d \to \{0,1\}$ is a Boolean function on $d \log q$-size inputs. Also, each $B \in \mathcal{T}$ can be described with $(d+1)m \log q$ bits, and so each function in the function family $\{f|_B\}_{B \in \mathcal{T}}$ has a short description.

Consider the problem: Given (a description of) $B \in \mathcal{T}$, construct a circuit that computes $f|_B$ well on average. We show the following.

Theorem III.A.6 (Hardness condensing)**.** *For an absolute constant $c > 0$, if a function f is δ-hard for $\mathsf{BPTIME}(t)//\log t$, then every probabilistic $t^{1/c}$-time algorithm $\mathcal{C}$ has probability at most $\epsilon = \max\{q^{-d/16}, t^{-1/c}\}$ (over random $B \in \mathcal{T}$ and the internal randomness of $\mathcal{C}$) of producing a circuit that $(1-\Omega(\delta))$-computes*

$f|_B$.

Intuitively, for almost every B, the function $f|_B$ has almost the same hardness as f, but is defined on inputs of smaller size. Thus the reduction from f to f_B can be thought of as "hardness condensing".

Finally, we can convert our uniform direct product theorem into a uniform version of the Yao XOR Lemma [Yao82]. While a qualitative version of this conversion follows immediately from [GL89], we obtain a quantitatively optimal version, up to constant factors. A uniform version of XOR Lemma is an approximate list decoding algorithm for a truncated version of the Hadamard code, and optimality is defined in terms of the information-theoretic coding properties of this code ([IJK06]). For $f : \{0,1\}^n \to \{0,1\}$ and $k \in \mathbb{N}$, the k-XOR encoding of f is the function $f^{\oplus k}$ mapping each k-subset of n-bit strings $(x_1, \dots, x_k)$ to the value $\oplus_{i=1}^{k} f(x_i)$.

Theorem III.A.7. *The k-XOR code is efficiently, locally, δ-approximately $(1/2+\epsilon, O(1/\epsilon^2))$-list decodable, for $\delta = O((\log 1/\epsilon)/k)$.*

III.A.4 Relation to previous work

Non-uniform Direct Product Theorem

The classical Direct-Product Theorem (and closely related Yao's XOR Lemma [Yao82]) for circuits has many proofs [Lev87, Imp95b, GNW95, IW97]. The basic idea behind all these proofs is the following: If a given circuit C' ϵ-computes $f^k(x_1, \dots, x_k)$, for some δ-hard function $f : \{0,1\}^n \to \{0,1\}$, with $\epsilon > (1-\delta)^k$, then it must be the case that the correctness of the answers of C' at some position i is *correlated* with the correctness of its answers in the remaining positions (since otherwise it would be the same as trying to compute $f(x_1), \dots, f(x_k)$ independently sequentially, which obviously cannot be done with probability greater than $(1-\delta)^k$).

This correlation of C''s answers can be exploited in various ways to get a circuit $(1-\delta)$-computing f from the circuit C' (yielding different proofs of the direct-product theorem in [Lev87, Imp95b, GNW95, IW97]). Usually, one takes a random k-tuple $(x_1, \dots, x_k)$ containing a given input x in some position i, runs C' on that tuple, and checks how well C' did in positions other than i. To perform such a check, one obviously needs to know the true values of f at the inputs x_j for $j \neq i$; these are provided in the form of non-uniform advice in the circuit model. Then one decides on the guess for the value $f(x)$ based on the quality of C''s answers for x_j, $j \neq i$. For example, in [IW97], one flips a random coin with probability that is some function of the number of incorrect answers given by C' outside position i.

Uniform Direct Product Theorem, and decoding vs. testing

To get a uniform algorithm for f, we need to remove (or at least minimize the amount of) the non-uniform advice $f(x_j)$, $j \neq i$. The first result of that type was obtained in [IJK06]. Their idea was to use the circuit C' itself in order to get enough labeled examples $(x, f(x))$, and then run the direct-product decoding algorithm of [IW97] on C' and the obtained examples.

To get sufficiently many examples, [IJK06] use a method they called direct product amplification, which is to take an algorithm solving the k-wise direct product to one that (approximately) solves the k'-wise direct product problem with $k' \gg k$. This amplification is essentially equivalent to approximate list decoding when there are only k' possible instances in the domain of the function f. Their list-decoding algorithm used one random "advice set" (where the algorithm produced correct answers) as a consistency check for another set that contains the instance to be solved. To be a meaningful consistency check, the advice set and instance-containing set need to have a large intersection. For independent random sets, this implies by the birthday-paradox bounds, that $k' \ll k^2$. Because of this constraint, [IJK06] had to use direct-product amplification iteratively, to cover the whole domain size of 2^n instances. These iterations complicated the construction and made the parameters far from optimal.

We instead pick the instance-containing set *conditioned* on having a large intersection with the advice set. This can be done at one shot, on any domain size, so no iterations are needed.

This idea is similar in spirit to the *direct-product testing* methods used by [GS00, DR06], and we were inspired by these papers. However, while they showed that this is sufficient in the unique decoding regime (where the algorithm is computing the direct product with high probability), we were surprised that this one idea sufficed in the list-decoding case as well. Our derandomized subspace construction was also inspired by [RS97, AS03], who list-decode functions correlated to multi-variable polynomials by using consistency checks on small dimensional subspaces.

While our results were inspired by similar results on direct-product testing, we have not found any formal connection between the testing and decoding problems. In particular, passing the consistency test with non-negligible probability is not sufficient to test non-negligible correlation with a direct-product function. It would be very interesting to find such a connection.

Remainder of the paper. Section III.B contains some background facts, and basic sampling properties of graphs used in decoding of intersection codes. The analysis of our algorithm $\mathcal{A}$ is given in Section III.C. where we state the conditions on the pair $(\mathcal{S}, \mathcal{T})$ that are sufficient for $\mathcal{A}$ to produce a good circuit $C_{A,v}$. Section III.D contains the proofs of Theorems III.A.4, III.A.5, and III.A.6. Theorem III.A.7 is proved in Section III.E. Section III.F contains concluding remarks and open questions.

III.B Preliminaries

III.B.1 Concentration bounds

The standard form of the Hoeffding bound [Hoe63] says that, for any finite subset F of measure α in some universe $\mathcal{U}$, a random subset R of size t is very likely to contain close to αt points from F. The following is a natural generalization for the case where F is any $[0,1]$-valued function over $\mathcal{U}$.

Lemma III.B.1 (Hoeffding [Hoe63]). *Let $F : \mathcal{U} \to [0,1]$ be any function over a finite universe $\mathcal{U}$ with the expectation $\mathbf{Exp}_{x\in\mathcal{U}}[F(x)] = \alpha$, for any $0 \le \alpha \le 1$. Let $R \subseteq \mathcal{U}$ be a random subset of size t. Define a random variable $X = \sum_{x\in R} F(x)$. Then the expectation of X is $\mu = \alpha t$, and for any $0 < \gamma \le 1$, $\mathbf{Pr}\left[|X-\mu| \ge \gamma\mu\right] \le 2 \cdot e^{-\gamma^2\mu/3}$.*

Lemma III.B.2. *Let $X_1, \dots, X_t$ be random variables taking values in the interval $[0,1]$, with expectations μ_i, $1 \le i \le t$. Let $X = \sum_{i=1}^t X_i$, and let $\mu = \sum_{i=1}^t \mu_i$ be the expectation of X. For any $0 < \gamma \le 1$, we have the following:*

- *[**Chernoff-Hoeffding**] If $X_1, \dots, X_t$ are independent, then $\mathbf{Pr}[|X-\mu| \ge \gamma\mu] \le 2 \cdot e^{-\gamma^2\mu/3}$.*
- *[**Chebyshev**] If $X_1, \dots, X_t$ are pairwise independent, then $\mathbf{Pr}[|X-\mu| \ge \gamma\mu] \le 1/(\gamma^2\mu)$.*

III.B.2 Pairwise independence of subspaces

Let $U = \mathbb{F}_q^m$ be an m-dimensional linear space over a finite field $\mathbb{F}_q$. An *affine* d-dimensional subspace A of U is specified by a collection of d linearly independent vectors $a_1, \dots, a_d \in U$ and an arbitrary vector $b \in U$ so that $A = \{b + \sum_{i=1}^d \alpha_i a_i \mid \alpha_i \in \mathbb{F}_q, 1 \le i \le d\}$. Thus the elements of A are in one-to-one correspondence with d-tuples of scalars $(\alpha_1, \dots, \alpha_d)$.

We will use the following easy fact.

Claim III.B.3. *A sequence of all q^d elements of a randomly chosen d-dimensional affine subspace of U are pairwise independent and uniform over U.*

A *linear* d-dimensional subspace A of U is specified by a collection of d linearly independent vectors $a_1, \ldots, a_d \in U$ so that $A = \{\sum_{i=1}^{d} \alpha_i a_i \mid \alpha_i \in \mathbb{F}_q, 1 \leq i \leq d\}$. It is no longer the case that *all* elements of a random linear subspace A are pairwise independent. For example, if vectors $\bar{\alpha} = (\alpha_1, \ldots, \alpha_d)$ and $\bar{\beta} = (\beta_1, \ldots, \beta_d)$ are scalar multiples of each other (i.e., are linearly dependent), then in every random subspace A the two corresponding elements of A will also be scalar multiples of each other.

However, if we restrict our attention to any sequence $\bar{\alpha}_1, \ldots, \bar{\alpha}_t$ of d-tuples $\bar{\alpha}_i \in \mathbb{F}_q^d$ such that every two of $\bar{\alpha}_i$'s are linearly independent, we get that the corresponding elements in a random d-dimensional linear subspace A are pairwise independent and uniform over U. It is easy to see that one can choose $t = (q^d - 1)/(q-1)$ nonzero vectors $\bar{\alpha}_1, \ldots, \bar{\alpha}_t \in \mathbb{F}_q^d$ such that every two of them are linearly independent. Thus we get the following.

Claim III.B.4. *For $t = (q^d - 1)/(q-1)$, let $\bar{\alpha}_1, \ldots, \bar{\alpha}_t \in \mathbb{F}_q^d$ be pairwise linearly independent vectors. Let A be a random d-dimensional linear subspace of U. Then the t vectors of A that correspond to $\bar{\alpha}_1, \ldots, \bar{\alpha}_t$ are pairwise independent and uniform over U.*

III.B.3 Graphs

We will consider bipartite graphs $G = G(L, R)$ defined on a bipartition $L \cup R$ of vertices; we think of L as left vertices, and R as right vertices of the graph G. For a vertex v of G, we denote by $N_G(v)$ the set of its neighbors in G; if the graph G is clear from the context, we will drop the subscript and simply write $N(v)$. We say that G is *bi-regular* if the degrees of vertices in L are the same, and the degrees of vertices in R are the same.

Auxiliary graphs for $(\mathcal{S}, \mathcal{T})$-codes

The following three graphs will be useful for the analysis of our intersection codes. Let U be any finite set. Let $\mathcal{T}$ be a family of k-subsets of U, and let $\mathcal{S}$ be a family of s-subsets of U, for some $s < k$.

Definition III.B.5 (inclusion graph)**.** The *inclusion graph* $I(\mathcal{S}, \mathcal{T})$ is the bipartite graph that has an edge (A, B) for every $A \in \mathcal{S}$ and $B \in \mathcal{T}$ such that $A \subseteq B$.

The inclusion graph $I(\mathcal{S}, \mathcal{T})$ is called *transitive* if, for every $B, B' \in \mathcal{T}$, there is a permutation π of U which moves B to B' and induces an isomorphism of I, and similarly, for every $A, A' \in \mathcal{S}$, there is a permutation σ of U which moves A to A' and induces an isomorphism of I.

Definition III.B.6 ($\mathcal{S}$-graph)**.** For every $B \in \mathcal{T}$, the $\mathcal{S}$-graph $H(B, N_I(B))$ is the bipartite graph that has an edge (x, A) for every $x \in B$ and $A \in N_I(B)$ such that $x \in A$.

Definition III.B.7 ($\mathcal{T}$-graph)**.** For every $A \in \mathcal{S}$, the $\mathcal{T}$-graph $G(U \setminus A, N_I(A))$ is the bipartite graph that has an edge (x, B) for every $x \in U \setminus A$ and $B \in N_I(A)$ such that $x \in B \setminus A$.

Note that if $I(\mathcal{S}, \mathcal{T})$ is transitive, then the structure of the $\mathcal{S}$-graph $H(B, N(B))$ is independent of the choice of B, and similarly, the structure of the $\mathcal{T}$-graph $G(U \setminus A, N(A))$ is independent of the choice of A. This will simplify the analysis of the properties of these graphs. One can easily check that the inclusion graph I for both of our running examples of families $(\mathcal{S}, \mathcal{T})$, Independent and Subspaces, is transitive.

Samplers

Let $G = G(L, R)$ be any bi-regular bipartite graph. For a function $\lambda : [0, 1] \to [0, 1]$, we say that G is a $(\mu, \lambda(\mu))$-*sampler* if, for every function $F : L \to$

$[0,1]$ with the average value $\mu \overset{\text{def}}{=} \mathbf{Exp}_{x\in L}[F(x)]$, there are at most $\lambda(\mu)\cdot|R|$ vertices $r \in R$ where

$$\left|\mathbf{Exp}_{y\in N(r)}[F(y)] - \mu\right| \geq \mu/2.$$

Note that the case of a Boolean function $F : L \to \{0,1\}$ with the average μ corresponds to the property that all but $\lambda(\mu)$ fraction of nodes $r \in R$ have close to the expected number of neighbors in the set $\{x \mid F(x) = 1\}$ of measure μ. The sampler defined above is a natural generalization to the case of $[0,1]$-valued F; it is also a special case of an *oblivious approximator* [BGG93] or *approximating disperser* [Zuc97].

For the analysis of intersection codes $Code(\mathcal{S},\mathcal{T})$ based on families $\mathcal{S}$ and $\mathcal{T}$, we will need that the corresponding $\mathcal{S}$-graphs and $\mathcal{T}$-graphs be samplers. We show that this is true for both of our running examples. Since both our inclusion graphs (for Independent and Subspaces cases) are transitive, the structure of the $\mathcal{S}$-graphs and $\mathcal{T}$-graphs is independent of the choices of $B \in \mathcal{T}$ and $A \in \mathcal{S}$, respectively.

Lemma III.B.8. *For both Independent and Subspaces families $(\mathcal{S},\mathcal{T})$, the $\mathcal{S}$-graph H is $(\alpha,\nu(\alpha))$-sampler, where*

- ***Independent:*** $\nu(\alpha) = 2\cdot e^{-\alpha k/24}$,
- ***Subspaces:*** $\nu(\alpha) = 4/(\alpha\sqrt{k})$.

Proof. For Independent, we use the Hoeffding bound of Lemma III.B.1. For Subspaces, we use the fact that points in a random affine subspace of a given affine space are uniformly distributed and pairwise independent (cf. Claim III.B.3), and then apply Chebyshev's bound of Lemma III.B.2. □

Lemma III.B.9. *For both Independent and Subspaces families $(\mathcal{S},\mathcal{T})$, the $\mathcal{T}$-graph G is $(\beta,\lambda(\beta))$-sampler, where*

- ***Independent:*** $\lambda(\beta) = 2\cdot e^{-\beta k/24}$,

- ***Subspaces:*** $\lambda(\beta) = 4q^2/(\beta\sqrt{k})$.

Proof. For Independent, we use the Hoeffding bound of Lemma III.B.1.

For subspaces, we use pairwise independence and the Chebyshev bound. Fix an affine subspace A of dimension r. Suppose A is $V + v$, for some r-dimensional linear subspace V of $U = \mathbb{F}_q^m$, and a vector $v \in U$. To choose a random $d = 2r$-dimensional affine subspace B containing A, we choose a random r-dimensional subspace W of U such that W is orthogonal to V, and define our affine $2r$-dimensional subspace $B = A + W$.

Note that all of $U \setminus A$ can be represented as the disjoint union of cosets $A+u$, over all distinct nonzero vectors u in the orthogonal subspace $V^{\perp}$. A function $F : (U \setminus A) \to [0,1]$ with the expectation β yields $[0,1]$-valued functions F_u where F_u is the restriction of F to the coset $A+u$, for every nonzero vector $u \in V^{\perp}$. Let β_u denote the average value of F_u over the points in $A+u$. Clearly, the average of β_u's is exactly β.

If we pick t nonzero vectors $u_1, \ldots, u_t \in V^{\perp}$ independently at random, we would obtain by the Chernoff-Hoeffding bound that the average $(1/t)\sum_{i=1}^{t} \beta_{u_i}$is very likely to be close to β. Similarly, if these t vectors were chosen pairwise independently, we could argue the concentration around the expectation β by Chebyshev's bound. The intuition is that vectors in a random r-dimensional subspace W are essentially pairwise independent, and hence we can argue that our random affine subspace B is likely to be a good sample for estimating the average of F.

More precisely, let $w_1, \ldots, w_t \in \mathbb{F}_q^r$ be any fixed collection of $t = (q^r - 1)/(q-1)$ nonzero vectors such that every two of them are linearly independent. By Claim III.B.4, in a random W the t corresponding vectors of W are pairwise independent and uniform over $V^{\perp}$. Let us denote by ω_i, $1 \le i \le t$, the element of W corresponding to w_i (i.e., ω_i is a linear combination of the basis vectors of W with scalar coefficients being the r field elements of w_i).

For each field element $i \in \mathbb{F}_q$, define $B_i = \cup_{j=1}^{t}(A + i \cdot \omega_j)$. Note that $B = \cup_{i \in \mathbb{F}_q} B_i$. Fix any nonzero $i \in \mathbb{F}_q$. For a random W, the vectors $i \cdot \omega_1, \ldots, i \cdot \omega_t$

are pairwise independent. By Chebyshev's bound of Lemma III.B.2, the probability that $(1/|B_i|) \cdot \sum_{x \in B_i} F(x)$ is less than $\beta/2$ or more than $3\beta/2$ is at most $4/(\beta t)$. By the union bound, the probability that at least one of B_i's deviates from the expectation is at most $4(q-1)/(\beta t)$. Thus, with probability at least $1 - 4/(\beta q^{r-2}) = 1 - 4q^2/(\beta s)$, a random affine subspace B containing A is a good sample for estimating the expectation of F. Since $s = \sqrt{k}$ for Subspaces, we get the desired claim. □

Properties of samplers and their subgraphs

Here we prove two properties of samplers, which will be useful for the analysis of our decoding algorithm. These properties basically show that samplers are "robust" to deletions of vertices.

The first property says that for any two large vertex subsets W and F of a sampler, the fraction of edges between W and F is close to the product of the densities of W and F.

Lemma III.B.10. *Suppose $G = G(L, R)$ is a (β, λ)-sampler. Let $W \subseteq R$ be any set of measure ρ, and let $F \subseteq L$ be any set of measure β. Then we have*

$$\mathbf{Pr}_{x \in L, y \in N(x)}[x \in F \ \& \ y \in W] \geq \beta(\rho - \lambda)/2.$$

Proof. We need to estimate the probability of picking an edge between F and W in a random experiment where we first choose a random $x \in L$ and then its random neighbor y. Since the graph G is assumed to be bi-regular, this probability remains the same in the experiment where we first pick a random $y \in R$ and its random neighbor $x \in N(y)$. The latter is easy to estimate using the sampling property of the graph G, as we show next.

Let $F' \subseteq F$ be of density exactly β. Let $W' \subseteq W$ be the subset of vertices that have at least $\beta/2$ fraction of their neighbors in F. Since G is a (β, λ)-sampler and W is of measure ρ, we get that W' is of measure at least $\rho - \lambda$. Then conditioned on picking a vertex $y \in W'$, the probability that its random neighbor is in F is at least $\beta/2$. The lemma follows. $\square$

The second property deals with edge-colored samplers. Suppose that all edges in a bi-regular graph $G = G(L, R)$ are colored with two colors, red and green, so that the number of red edges is at most t, for some $t \geq 0$. Since G is bi-regular, picking a random vertex $x \in L$ and its random incident edge is the same as picking a random $y \in R$ and its random incident edge, and clearly, the probability of getting a red edge in both cases is $t/|E|$, where E is the edge set of G. Now suppose that we are given a subgraph G' obtained from G by removing some vertices from R (and all the edges incident upon the removed vertices). Let $W \subseteq R$ be a subset of the remaining vertices in G', and suppose that G' has at most t red edges. Since G' is still right-regular (i.e., all vertices $w \in W$ have the same degree), sampling a random incident edge of a random vertex $w \in W$ still yields a red edge with probability at most $t/|E'|$, where E' is the edge set of G'. For general graphs G, we can't say that the probability of getting a red edge remains the same when we pick a random incident edge of a random vertex $x \in L$ (since G' may not be bi-regular). However, we prove that this is approximately true when G is a sampler.

Lemma III.B.11. *Suppose $G = G(L, R)$ is a (β, λ)-sampler, with the right degree D. Let $W \subseteq R$ be any subset of density ρ, and let $G' = G(L, W)$ be the induced subgraph of G (obtained after removing all vertices in $R \setminus W$), with the edge set E'. Let $Col : E' \to \{red, green\}$ be any coloring of the edges of G' such that at most $\alpha D|W|$ edges are colored red, for some $0 \leq \alpha \leq 1$. Then*

$$\mathbf{Pr}_{x \in L, y \in N_{G'}(x)}[Col(\{x, y\}) = red] \leq \max\{2\alpha/(1 - \lambda/\rho), \beta\}.$$

Proof. We need to estimate the probability of picking a red edge in G' when we first pick a random $x \in L$ and then pick its random neighbor y in G'. For every $x \in L$, let d_x be the degree of x in G', and let $\xi(x)$ be the fraction of red edges incident to x in G'. The probability we want to estimate is exactly $\mu = \mathbf{Exp}_{x\in L}[\xi(x)]$. If $\mu \leq \beta$, then we are done. So for the rest of the proof, we will suppose that $\mu > \beta$.

Let $W' \subseteq W$ be the subset of those vertices w where $\mathbf{Exp}_{x\in N(w)}[\xi(x)] \geq \mu/2$. (Here we use $N(w)$ to denote the neighborhood $N_{G'}(w)$ of w in G', which is the same as $N_G(w)$ by the definition of G'.) Since G is a (β,λ)-sampler and W has measure ρ in G, we get that W' has measure at least $\rho - \lambda$ in G, and hence measure $1 - \lambda/\rho$ in G'. Hence, we have

$$\sum_{y\in W} \mathbf{Exp}_{x\in N(y)}[\xi(x)] \geq \sum_{y\in W'} \mathbf{Exp}_{x\in N(y)}[\xi(x)] \geq |W|(1-\lambda/\rho)\mu/2. \tag{III.1}$$

On the other hand, $\sum_{y\in W} \left(D \cdot \mathbf{Exp}_{x\in N(y)}[\xi(x)]\right)$ is simply the summation over all edges (x,y) in G' where each edge (x,y) with $x \in L$ contributes $\xi(x)$ to the sum. Since the degree of each x is d_x, each $x \in L$ contributes exactly $d_x\xi(x)$, which is the number of incident red edges at x. Hence, the total sum is exactly the number of red edges in G', which is at most $\alpha D|W|$ by assumption. It follows that

$$\sum_{y\in W} \mathbf{Exp}_{x\in N(y)}[\xi(x)] = (1/D)\sum_{x\in L} d_x\xi(x) \leq |W|\alpha. \tag{III.2}$$

Finally, comparing the bounds in Eqs. (V.1) and (V.2), we conclude that $\mu \leq 2\alpha/(1-\lambda/\rho)$. □

III.C Decoding intersection codes

Let $(\mathcal{S},\mathcal{T})$ be a pair of families of subsets of U, and let $Code(\mathcal{S},\mathcal{T})$ be the intersection code defined for these families. Fix a function $f : U \to R$. Let C' be a circuit that ϵ-computes $Code(f)$. We will show how to compute from C' a deterministic circuit C that $(1-\delta)$-computes f, for $\delta > 0$ being the parameter that depends on ϵ and $(\mathcal{S},\mathcal{T})$.

Our decoding algorithm $\mathcal{A}$ for $Code(\mathcal{S},\mathcal{T})$ can be defined in terms of the inclusion and $\mathcal{T}$-graphs. Fix any edge (A,B) of the inclusion graph $I(\mathcal{S},\mathcal{T})$. Let $v = C'(B)|_A$ be the values that the circuit $C'(B)$ gives for the elements in A.

Let $G = G(U \setminus A, N(A))$ be the $\mathcal{T}$-graph for A. Let $Cons \subseteq N(A)$ be the subset of those $B' \in N(A)$ for which $C'(B')|_A = v$. We will say that such sets B' are *consistent with* B.

Define the **circuit $\mathbf{C_{A,v}}$**:

"On input $x \in U$, if $x \in A$, then output the corresponding value v_x. Otherwise, repeatedly sample random neighbors B' of x in the $\mathcal{T}$-graph G, discarding any $B' \notin Cons$, until the first $B' \in Cons$ is obtained. For this $B' \in Cons$, output the value $C'(B')|_x$. Produce the default (error) output if no $B' \in Cons$ is found even after $O((\ln 1/\delta)/\epsilon)$ steps."

Define the **decoding algorithm $\mathcal{A}$**:

"On an input circuit C', pick a random edge (A, B) of the inclusion graph $I(\mathcal{S}, \mathcal{T})$, set $v = C'(B)|_A$, and output the circuit $C_{A,v}$."

Remark III.C.1. For the described algorithm $C_{A,v}$ to be efficient, we need an efficient procedure for sampling random neighbors of a given left vertex in the $\mathcal{T}$-graph $G(U \setminus A, N(A))$. For both of our running examples, one can easily argue that such efficient sampling is possible.

We now state the main technical result of our paper: the conditions on $(\mathcal{S}, \mathcal{T})$ under which the decoding algorithm $\mathcal{A}$ produces a good circuit $C_{A,v}$. For the rest of this section, we set $\epsilon' = \epsilon/2$.

Theorem III.C.2. *Suppose that the inclusion graph $I(\mathcal{S}, \mathcal{T})$ is transitive (and hence also bi-regular), the $\mathcal{S}$-graph H is a $(\mu, \delta\epsilon'^2/(128\mu))$-sampler for every $\mu > \delta/64$, and the $\mathcal{T}$-graph G is a $(\delta/16, \epsilon'/2)$-sampler. Then the algorithm $\mathcal{A}$ produces with probability $\epsilon'/2$ a randomized circuit $C_{A,v}$ satisfying*

$$\mathbf{Pr}[C_{A,v} \text{ computes } f] \geq 1 - \delta/4,$$

where the probability is over the inputs and the internal randomness of $C_{A,v}$.

Remark III.C.3. Note that if a randomized circuit $C_{A,v}$ satisfies the conclusion of Theorem III.C.2, then by randomly fixing its internal randomness we get (with probability at least 3/4) a *deterministic* circuit C that $(1-\delta)$-computes f.

We postpone the proof of Theorem III.C.2, and use it to prove Theorem III.A.3.

Proof of Theorem III.A.3. For Independent, we get by Lemmas III.B.8 and III.B.9 that both H and G are $(\mu, \lambda(\mu))$-samplers for $\lambda(\mu) \leq e^{-\Omega(\mu k)}$. For $\mu > \delta/64$, write $\mu = c\delta$ where $c = \mu/\delta > 1/64$. For the graph H, we get that $\mu \cdot \lambda(\mu) \leq c\delta e^{-\Omega(c\delta k)}$. For $\delta = d\log(1/\epsilon)/k$ for large enough constant d, we get $e^{-\Omega(cd \log 1/\epsilon)} = \epsilon^{\Omega(cd)} \leq \epsilon'^2 \epsilon^{cd'}$, for some large constant d' dependent on d. Assume that $\epsilon < 0.9$ (if a circuit C' ϵ-computes f^k for $\epsilon \geq 0.9$, it obviously 0.9-computes f^k).[3] Choosing sufficiently large constant d, we can ensure that $\epsilon^{cd'} < 2^{-c}/128$, and so $c\delta e^{-\Omega(c\delta k)} \leq c\delta\epsilon'^2 2^{-c}/128 \leq \delta\epsilon'^2/128$. Thus H satisfies the assumptions of Theorem III.C.2. Setting $\delta = d(\log 1/\epsilon)/k$ for a large enough $d \in \mathbb{N}$ will also make the $\mathcal{T}$-graph G satisfy the assumptions of of Theorem III.C.2.

For Subspaces, Lemma III.B.8 gives us that H is $(\mu, \lambda(\mu))$-sampler for $\lambda(\mu) = 4/(\mu\sqrt{k})$. Hence, $\mu \cdot \lambda(\mu) \leq 4/\sqrt{k}$. The latter is at most $\delta\epsilon'^2/128$ for $\delta \geq 512/\epsilon'^2\sqrt{k}$. Lemma III.B.9 says that the graph G is $(\delta/16, \epsilon'/2)$-sampler for $\delta \geq 128q^2/(\epsilon'\sqrt{k})$. Thus, to satisfy the conditions of Theorem III.C.2, we can set $\delta \leq 512q^2/(\epsilon'^2\sqrt{k}))$, which is $O(1/(\epsilon'^2 k^{1/4}))$ for $q \leq k^{1/8}$.

By Remark III.C.3, we get in both cases a required deterministic circuit $(1-\delta)$-computing f. $\square$

III.C.1 Why $C_{A,v}$ works

Here we describe the conditions on our auxiliary graphs (inclusion, $\mathcal{S}$- and $\mathcal{T}$-graphs) and an edge (A, B) of the inclusion graph, which are sufficient

[3]In fact, if a circuit C' ϵ-computes f^k for $\epsilon \geq 0.9$, then for $k > 1/\delta$, there is a *single* algorithm that $(1-\delta)$-computes f: "Given input x, sample $O(\log 1/\delta)$ random k-sets B containing x, and output the majority answer of $C'(B)|_x$." For the analysis, it suffices to show that for each but $\delta/2$ fraction of inputs x, there are at least 2/3 sets B containing x such that $C'(B) = f^k(B)$, which is easy to argue.

for the circuit $C_{A,v}$ described above to satisfy the conclusion of Theorem III.C.2. Intuitively, we are using (A, v) as a consistency check to see whether to believe $C'(B')$. To be useful as a consistency check, we should have:

- $v = f(A)$, so if $C'(B')$ is correct, it will always be consistent with v on A.
- There are many B' for A where $C'(B')$ is correct.
- On average over B' where $C'(B')$ is consistent with A, $C'(B')$ is correct for most $x \in B' \setminus A$.

We show that these conditions suffice, and that many such sets A exist.

We need the following definitions. For a set $B \in \mathcal{T}$, let $Err(B)$ denote the subset of those x's in B where $C'(B)$ disagrees with $f^k(B)$, and let $err(B) = |Err(B)|/|B|$. A set $B \in \mathcal{T}$ is called *correct* if $err(B) = 0$. A set $B \in \mathcal{T}$ is called *α-incorrect* if $err(B) \geq \alpha$. For the inclusion graph $I(\mathcal{S}, \mathcal{T})$, we call an edge (A, B) *correct* if B is correct. As before, we set $\epsilon' = \epsilon/2$. Call an edge (A, B) *good* if it is correct and at least ϵ'-fraction of all edges (A, B') incident to A are correct. An edge (A, B) of the inclusion graph is called *α-excellent* if it is *good*, and moreover,

$$\mathbf{Exp}_{B' \in Cons}[err(B')] \leq \alpha,$$

where the expectation is over uniformly random B' that are consistent with B.

In words, for an excellent edge (A, B), we have at least ϵ' of correct edges (A, B') (and so these $B' \in Cons$), and at the same time, the average fraction of errors in the neighbors of A that are consistent with B is less than α. So, conditioned on sampling a random $B' \in Cons$, we expect to get a B' such that $C'(B')|_x = f(x)$ for most $x \in B'$.

Our circuit $C_{A,v}$ is defined so that it only considers random $B' \in Cons$. This circuit will agree with f well on average, assuming that A, v came from some excellent edge (A, B), and assuming that the $\mathcal{T}$-graph is a sampler.

Lemma III.C.4. *Let an edge (A, B) of the inclusion graph I be α-excellent, and let the $\mathcal{T}$-graph $G(U \setminus A, N(A))$ be a (β, λ)-sampler. Suppose that $\lambda \leq \epsilon'/2$, $\alpha \leq \beta/2$,* *and $\beta \leq \delta/16$. Then $\mathbf{Pr}[C_{A,v}$ computes $f] \geq 1 - \delta/4$, where the probability is over uniform x's and the internal randomness of $C_{A,v}$.*

To prove Lemma III.C.4, we consider two cases. First we consider the set $F \subseteq U \setminus A$ of x's that have too few edges (x, B') with $B' \in Cons$ in the $\mathcal{T}$-graph $G(U \setminus A, N(A))$. These are the x's for which $C_{A,v}$ is unlikely to produce any answer and hence fails. Secondly, we bound the average conditional probability of $C_{A,v}$ producing an incorrect answer given that the circuit produces some answer. Note that for every $x \in U \setminus A$ this conditional probability is the same for all sampling steps of $C_{A,v}$. So, we can just analyze this conditional probability for one sampling step.

First, we bound the size of F.

Lemma III.C.5. *Suppose an edge (A, B) of I is good, and the $\mathcal{T}$-graph $G(U \setminus A, N(A))$ is a (β, λ)-sampler. Let F be the subset of $U \setminus A$ with less than μ fraction of their edges into Cons, where $\mu = (\epsilon' - \lambda)/2$. Then the measure of F is at most β.*

Proof. Suppose that F has density at least β. Let $F' \subseteq F$ be of density exactly β. By the assumption of the lemma, we have that $\mathbf{Pr}_{x \in U \setminus A, y \in N(x)}[x \in F' \ \&\ y \in Cons] < \beta\mu = \beta(\epsilon' - \lambda)/2$.

On the other hand, we know that $Cons$ has density at least ϵ' (by the definition of goodness of (A, B)). By Lemma V.E.2, the fraction of edges in G that go between F and $Cons$ is at least $\beta(\epsilon' - \lambda)/2$, which contradicts our earlier upper bound. □

For a given $x \in U \setminus A$, let $h(x)$ denote the conditional probability that $C_{A,v}$ produces an incorrect answer, given that it produces some answer. We will show that the expectation $\mathbf{Exp}_{x \in U \setminus A}[h(x)]$ is small.

Lemma III.C.6. *Suppose (A, B) is α-excellent, and the $\mathcal{T}$-graph G is a (β, λ)-sampler. Further suppose that $\alpha \leq \beta/2$ and $\lambda \leq \epsilon'/2$. Then $\mathbf{Exp}_{x \in U \setminus A}[h(x)] \leq \beta$.*

Proof. Since $C_{A,v}$ produces an answer on a given input x only if it samples a consistent neighbor B' of x in the $\mathcal{T}$-graph $G(U \setminus A, N(A))$, we can view $h(x)$ as follows. Let $G' = G(U \setminus A, Cons)$ be the induced subgraph of G where we remove all inconsistent vertices from $N(A)$. For each edge (x, B') of G', we color it red if $x \in Err(B')$, and color it green otherwise. Then $h(x)$ is the fraction of red edges incident to x in the graph G'.

Let ρ be the measure of $Cons$ in G. We know that $\rho \geq \epsilon'$. Let $D = |B|$ be the right degree of the $\mathcal{T}$-graph G (and hence also of G'). The total number of red edges in G' is at most $\alpha D|Cons|$, by the definition of α-excellence.

By Lemma V.E.3, we conclude that $\mathbf{Pr}_{x \in U \setminus A, B' \in N_{G'}(x)}[x \in Err(B')] \leq \max\{2\alpha/(1 - \lambda/\rho), \beta\}$. By assumptions, $1 - \lambda/\rho \geq 1 - \lambda/\epsilon' \geq 1/2$, and so $\alpha/(1 - \lambda/\epsilon') \leq 2\alpha \leq \beta$. □

Now we can finish the proof of Lemma III.C.4.

Proof of Lemma III.C.4. Lemma III.C.5 implies for every $x \in U \setminus (A \cup F)$, where F is of measure at most β, there are at least $\epsilon'/4$ fraction of edges into $Cons$. Hence the probability of $C_{A,v}$ not producing any answer in $t = d(\log 1/\delta)/\epsilon'$ sampling steps for such an x is at most $\delta/8$ for some constant d, e.g., $d = 100$. For each such x, the probability that $C_{A,v}$ is wrong, given that $C_{A,v}$ produces an answer, is $h(x)$. Hence, the overall probability (over random x and internal randomness) that $C_{A,v}$ is wrong is at most $\beta + \delta/8 + \mathbf{Exp}_{x \in U \setminus A}[h(x)]$. By Lemma III.C.6, the last summand is at most β, and so the total is at most $2\beta + \delta/8 \leq \delta/4$ (since $\beta \leq \delta/16$). □

III.C.2 Choosing an excellent edge (A, B)

Here we show that if the inclusion graph I is bi-regular and if the $\mathcal{S}$-graph H is a sampler, then a random edge (A, B) of I will be excellent with probability $\Omega(\epsilon)$.

Lemma III.C.7. *Suppose the inclusion graph I is bi-regular, and the $\mathcal{S}$-graph H is an $(\mu, \nu(\mu))$-sampler*[4]*. Moreover, assume that $0 \leq \alpha \leq 1$ is such that, for every $\alpha/2 < \mu \leq 1$, we have $\mu \cdot \nu(\mu) \leq \alpha\epsilon'^2/4$. Then a random edge (A, B) of I is α-excellent with probability at least $\epsilon'/2$.*

First, we argue the following.

Lemma III.C.8. *A random edge (A, B) of a bi-regular inclusion graph I is good with probability at least ϵ'.*

Proof. Choosing a random edge (A, B) of the inclusion graph I is equivalent to choosing a random $B \in \mathcal{T}$ and then choosing a random $A \in N(B)$. By the assumption on C', a random $B \in \mathcal{T}$ is correct with probability at least ϵ. Thus we have $\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[(A, B) \text{ is correct}] \geq \epsilon$.

For $A \in \mathcal{S}$, let $P(A)$ be the event (over a random choice of $A \in \mathcal{S}$) that $\mathbf{Pr}_{B' \in N(A)}[B' \text{ is correct}] < \epsilon/2$. Observe that, conditioned on $A \in \mathcal{S}$ such that $P(A)$, we get

$$\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[(A, B) \text{ is correct} \mid P(A)] < \epsilon/2,$$

and so,

$$\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[((A, B) \text{ is correct}) \;\&\; P(A)] < \epsilon/2.$$

Finally, the probability that a random edge (A, B) is good is equal to

$$\mathbf{Pr}_{A,B}[(A, B) \text{ is correct}] - \mathbf{Pr}_{A,B}[((A, B) \text{ is correct}) \;\&\; P(A)] > \epsilon - \epsilon/2 = \epsilon/2,$$

which is equal to ϵ', as required. $\square$

Now we can prove Lemma III.C.7.

Proof of Lemma III.C.7. To show that an edge (A, B) is α-excellent, it suffices to argue that

$$\sum_{B' \in Cons:\ err(B') > \alpha/2} err(B') \leq (\alpha/2)|Cons|,$$

[4] *Here we only need that, for any measure μ subset F of left vertices of H, the fraction of right vertices with no incident edges into F is at most ν.*

where $Cons$ is the set of all $B' \in N(A)$ that are consistent with B. This expression can be equivalently rewritten as

$$\mathbf{Pr}_{B' \in Cons, x \in B'}[err(B') > \alpha/2 \; \& \; x \in Err(B')] \leq \alpha/2. \tag{III.3}$$

For independent random $A \in \mathcal{S}$ and $B \in N(A)$, let $E_1(A, B)$ be the event that (A, B) is good, but the inequality (III.3) does not hold (i.e., the probability in (III.3) is greater than $\alpha/2$).

For independent random $A \in \mathcal{S}$, $B \in N(A)$, $B' \in N(A)$, and $x \in B'$, let $E(A, B, B', x)$ be the event that

$$(A, B) \text{ is correct } \; \& \; B' \in Cons \; \& \; err(B') > \alpha/2 \; \& \; x \in Err(B').$$

The probability of E is the average over all $B' \in \mathcal{T}$ of the conditional probabilities of E given B'. Consider any fixed B' with $err(B') > \alpha/2$. For each such B', the set A is a uniform element of $N(B')$ in the inclusion graph. By the sampling property of the $\mathcal{S}$-graph $H(B', N(B'))$, the probability that a random $A \in N(B')$ completely misses the subset $Err(B')$ is at most $\nu(err(B'))$. If A has nonempty intersection with $Err(B')$, then it cannot be the case that both (A, B) is correct and $B' \in Cons$. Hence, given B', the conditional probability of the event E is at most $\nu(err(B')) \cdot err(B')$, and so,

$$\mathbf{Pr}[E] \leq \frac{1}{|\mathcal{T}|} \sum_{B' \in \mathcal{T} : err(B') > \alpha/2} err(B') \cdot \nu(err(B')),$$

which is at most $\alpha\epsilon'^2/4$ by the assumption of the lemma.

We have

$$\mathbf{Pr}[E \mid E_1] > (\alpha/2)\mathbf{Pr}_{B' \in \mathcal{T}}[B' \in Cons \mid E_1] \geq \alpha\epsilon'/2, \tag{III.4}$$

where the first inequality is by the definition of the event E_1, and the second inequality by the definition of goodness of (A, B). On the other hand, $\mathbf{Pr}[E \mid E_1] = \mathbf{Pr}[E \; \& \; E_1]/\mathbf{Pr}[E_1] \leq \mathbf{Pr}[E]/\mathbf{Pr}[E_1]$. Combined with (III.4), this implies that $\mathbf{Pr}[E_1] \leq \mathbf{Pr}[E] \cdot 2/(\alpha\epsilon') \leq \epsilon'/2$.

Clearly, $\mathbf{Pr}_{A\in\mathcal{S},B\in N(A)}[(A,B)$ is α-excellent] is at least

$$\mathbf{Pr}_{A\in\mathcal{S},B\in N(A)}[(A,B) \text{ is good}] - \mathbf{Pr}_{A\in\mathcal{S},B\in N(A)}[E_1].$$

By Lemma III.C.8, the first probability in the difference above is at least ϵ', and, by what we showed earlier, the second probability is at most $\epsilon'/2$. The lemma follows. □

Proof of Theorem III.C.2. The proof follows easily from Lemmas III.C.4 and III.C.7. We simply set $\beta = \delta/16$, $\lambda = \epsilon'/2$, $\alpha = \beta/2 = \delta/32$, and $\nu(\mu) = \alpha\epsilon'^2/(4\mu) = \delta\epsilon'^2/(128\mu)$. □

III.D Extensions

III.D.1 Approximate version of the Uniform Direct-Product Theorem

In this section, we prove Theorem III.A.4. The proof is along the same lines as that of Theorem III.A.3 given in the previous section. We just need to make the following modifications in our definitions. Before, if $C'(B)$ was correct, it was correct on the subset A. Here, we need to bound the chance that, even if $C'(B)$ is almost correct, its number of mistakes on A is disproportionately high. We include this in the definition of "correct edge", so that two correct edges for A will be (mostly) consistent on A. Second, before, we had the correct values for A, and any deviation from these values could be used to rule out $C'(B')$ as inconsistent. Now, our values for even good A and B' are somewhat faulty, and so could be somewhat inconsistent. We need to redefine consistency to allow a small number of contradictory values, and then show that any very incorrect $C'(B')$ will have too many inconsistent values with high probability.

Recall that for $B \in \mathcal{T}$, $Err(B)$ is the set of those $x \in B$ where $C'(B)|_x \neq f(x)$, and $err(B) = |Err(B)/|B|$. We say that a set $B \in \mathcal{T}$ is *δ'-correct* if $err(B) \leq \delta'$ (i.e., $C'(B)$ and $f^k(B)$ disagree on at most δ' fraction of elements of B). An edge (A, B) of the inclusion graph I is called *δ'-correct* if B is δ'-correct and $|A \cap Err(B)| \leq 2\delta'|A|$.

For this section, we set $\epsilon' = \epsilon/4$. Call an edge (A, B) of I *good* if it is δ'-correct and at least ϵ'-fraction of all neighbors B' of A are δ'-correct.

The definition of consistency changes as follows. Two neighbors B, B' of A are called *consistent* if $C'(B)|_A$ and $C'(B')|_A$ disagree on at most $4\delta'$ fraction of elements in A. Note that for any two δ'-correct edges (A, B) and (A, B'), we have that B and B' are consistent. As before, for a given edge (A, B), we denote by $Cons$ the set of all B' that are consistent with B. Finally, the definition of an excellent edge is as before: An edge (A, B) is *α-excellent* if it is good, and moreover, $\mathbf{Exp}_{B' \in Cons}[err(B')] \leq \alpha$.

Next we need to verify that with these modifications in the definitions, all lemmas of the previous section go through. It is straightforward to check that all lemmas in Section III.C.1 continue to hold (with the same proofs) with respect to these new definitions.

For lemmas of Section III.C.2, we need to argue that a random edge (A, B) is excellent with probability $\Omega(\epsilon)$. For this, we need an analogue of Lemma III.C.8.

Lemma III.D.1. *Suppose the inclusion graph I is bi-regular, and the $\mathcal{S}$-graph H is $(\delta', 1/2)$-sampler. Then a random edge (A, B) of the inclusion graph I is good with probability at least ϵ'.*

Proof. We choose a random edge (A, B) of I by choosing a random $B \in \mathcal{T}$ first, and choosing a random $A \in N(B)$. By the assumption on the circuit C', the probability that a random $B \in T$ is δ'-correct is at least ϵ. For every fixed δ'-correct set B, the sampling property of the $\mathcal{S}$-graph implies that $\mathbf{Pr}_{A \in N(B)}[|A \cap Err(B)| > 2\delta'|A|] \leq 1/2$. It follows that a random edge (A, B) is δ'-correct with probability at least $\epsilon/2$.

Similarly to the proof of Lemma III.C.8, let $P(A)$ be the event that

$\mathbf{Pr}_{B' \in N(A)}[(A, B')$ is δ'-correct$] < \epsilon/4$. We get that

$$\mathbf{Pr}_{A \in \mathcal{S}, B \in N(A)}[((A, B) \text{ is } \delta'\text{-correct}) \ \& \ P(A)] < \epsilon/4.$$

Finally, the probability that (A, B) is good is equal to the probability that it is δ'-correct, minus the probability that it is δ'-correct and the event $P(A)$ happens. The former is $\epsilon/2$, and the latter is is less than $\epsilon/4$. Thus (A, B) is good with probability at least $\epsilon/4$, as required. □

We have the following analogue of Lemma III.C.7.

Lemma III.D.2. *Suppose the inclusion graph I is bi-regular, and the $\mathcal{S}$-graph H is $(\mu, \nu(\mu))$-sampler. Assume that $1 \geq \alpha \geq 24\delta'$ is such that for every $1 \geq \mu > \alpha/2$, $\mu \cdot \nu(\mu) < \alpha\epsilon'^2/4$. Then a random edge (A, B) of I is α-excellent with probability at least $\epsilon'/2$.*

Proof sketch. Compared with the proof of Lemma III.C.7, the only change is in the argument to upperbound $\mathbf{Pr}[E(A, B, B', x)]$. This is modified as follows. Condition on any set $B' \in \mathcal{T}$ that is μ-incorrect, for $\mu > \alpha/2$. By the sampling property of the $\mathcal{S}$-graph, the probability that a random neighbor $A \in N(B')$ has less than $\mu/2$ fraction of elements from $Err(B')$ is at most $\nu(\mu)$. Consider any fixed A that has more than $\mu/2$ fraction of elements from $Err(B')$. For any neighbor B of A such that B is consistent with B', we have that A contains more than $(\mu/2 - 4\delta')|A|$ elements from $Err(B)$, which is more than $2\delta'|A|$ for $\mu > \alpha/2 \geq 12\delta'$, and so the edge (A, B) is not δ'-correct. This implies that the conditional probability $\mathbf{Pr}[E(A, B, B', x) \mid B'] \leq \mu \cdot \nu(\mu)$. The rest of the proof is exactly the same as that of Lemma III.C.7. □

With the lemmas above, we get the proof of Theorem III.A.4 in the same way as the proof of Theorem III.A.3, for $\delta' \geq \Omega(\delta)$.

III.D.2 Derandomized Direct-Product Theorems

Here we will prove Theorem III.A.5. For $K = \text{poly}(1/\epsilon)$ and $k = O(\log 1/\epsilon)$, let $\mathcal{K}$ denote the collection of all k-subsets of $\{1, \ldots, K\}$. We need to analyze the function $h : \mathcal{T} \times \mathcal{K} \to \{0,1\}^k$ mapping $(T, i_1, \ldots, i_k)$ to $g(T)|_{i_1,\ldots,i_k}$, where $\mathcal{T}$ is a collection of affine d-dimensional subspaces of $\mathbb{F}_q^m$.

First we analyze the input size of h. It consists of $O(n)$ bits to describe a constant-dimensional affine subspace T, plus $k \log K = O((\log 1/\epsilon)\delta^{-1} \cdot (\log 1/\epsilon + \log 1/\delta)) = O((\log 1/\epsilon)^2)$ bits to specify the k-subset of $\{1, \ldots, K\}$, for constant δ. For $\epsilon \geq e^{-\Omega(\sqrt{n})}$, we get that the total input size is $O(n)$.

Suppose h is ϵ-computable in $\mathsf{BPTIME}(t^{1/c})//(1/c)\log t$. Given a circuit ϵ-computing h, we will show how to efficiently compute a list of circuits one of which $(1-\delta)$-computes f. This will imply that f is $(1-\delta)$-computable in $\mathsf{BPTIME}(t)//\log t$, contrary to the assumption of the theorem.

Our argument follows along the lines of a standard analysis of code concatenation (see, e.g., [STV01]). Suppose we have a circuit C' that ϵ-computes h. By averaging, we get that for at least $\epsilon/2$ fraction of $T \in \mathcal{T}$, the equality $C'(T, \kappa) = g(T)|_\kappa$ holds for at least $\epsilon/2$ fraction of k-subsets $\kappa \in \mathcal{K}$. Call $\mathcal{T}_{good}$ the set of such good Ts.

By Theorem III.A.3, we know that the Independent intersection code is δ'-approximately $(\epsilon/2, O(1/\epsilon))$-list decodable. So, for every $T \in \mathcal{T}_{good}$, we can efficiently recover a list of $\ell = O(1/\epsilon)$ length-K strings, one of which $(1-\delta')$-agrees with $g(T)$.

For each $T \in \mathcal{T}$, let us order the strings returned by our approximate list-decoding algorithm on input $C'(T, \cdot)$. Define a list of ℓ circuits $C''_1, \ldots, C''_\ell$ for $g(T)$, where $C''_i(T)$ outputs the ith K-bit string on the list corresponding to T. By averaging, there is some $1 \leq i \leq \ell$ such that $C''_i(T)$ will $(1-\delta')$-agree with $g(T)$ for at least $1/\ell$ fraction of inputs $T \in \mathcal{T}_{good}$, which is at least $\Omega(\epsilon^2)$ fraction of all inputs T to g. Let us call such a circuit C''_i approximately good for g.

By Theorem III.A.4, the Subspaces intersection code is (δ, δ') - approximately $(\Omega(\epsilon^2), O(1/\epsilon^2))$-list-decodable. Thus, for each of our ℓ circuits $C''_1, \ldots, C''_\ell$, we efficiently get $O(1/\epsilon^2)$ new circuits such that, if C''_i is an approximately good circuit for g, then the list of circuits obtained from that C''_i will have a circuit

$(1-\delta)$-computing f. Overall, we efficiently construct a list of $O(\ell/\epsilon^2) = O(1/\epsilon^3)$ circuits for f, one which will $(1-\delta)$-compute f. Hence, f is not δ-hard for $\mathsf{BPTIME}(t)//\log t$. A contradiction.

III.D.3 Hardness condensing

In this subsection, we reinterpret the results of the previous section to give a version of hardness condensing for the semi-uniform model, proving Theorem III.A.6.

Imagine the sets B before as being exponentially large but succinctly representable (as in the subspace construction for large values of $k = q^d$). The idea is that, instead of $C'(B)$ explicitly giving the values of f on B, we could replace $C'(B)$ with a meta-algorithm that produces a circuit that computes $f|_B$. We could still estimate the agreement of two such circuits on A. Thus, if f is hard, the restricted function $f|_B$ is hard for almost all B.

To get a precise statement of this idea, consider a family of functions $f_h(x) = F(h,x)$ where $h \in \mathcal{H}$ and $x \in U = \{0,1\}^n$. Call the family f_h $(1-\epsilon)$-hard to δ-compute in semi-uniform time t if, for any time $t(|h|+n)$ probabilistic algorithm $A(h,r)$ which produces a circuit $C_{h,r}$ on n bit inputs x, we have $\mathbf{Pr}_{h,r}[C_{h,r}\ \delta\text{-computes } f_h] \leq \epsilon$. [5]

Assume $\mathcal{S}$ and $\mathcal{T}$ meet the conditions of Theorem III.C.2, and that furthermore, we can describe $B \in \mathcal{T}$ and $A \in \mathcal{S}$ as strings of length n_1, and sample uniformly from either, using at most n_2 random bits, in time polynomial in n_1+n_2. For x of length n_2, let $f_B(x)$ be f applied to the random element of B obtained by using string x in the sampling algorithm. (For example, in the case B is a d-dimensional affine subspace of $(\mathbb{F}_q)^m$, we can represent B by its basis and skew vectors $b_1, \ldots, b_d, v$, with $n_1 = (d+1)m\log q$ bits. Then with $n_2 = d\log q$ bits, we can sample from B by picking random $\alpha_1, \ldots, \alpha_d$ and letting $y = \alpha_1 b_1 + \cdots + \alpha_d b_d + v$.

[5] Note that this definitions is a generalization of hardness of a single function in the semi-random model: f being δ-hard for $\mathsf{BPTIME}(t(n))//l(n)$ is the same as the function family with single member f being $(1-1/2^{l(n)})$-hard to $(1-\delta)$-compute in semi-uniform time $t(n)$.

Then $f_{b_1,\ldots,b_d,v}(\alpha_1,\ldots,\alpha_d) = f(\alpha_1 b_1 + \cdots + \alpha_d b_d + v)$.)

Then by altering the previous proof of Theorem III.A.4 as specified above, we have:

Theorem III.D.3. *Let $\mathcal{S}$, $\mathcal{T}$, δ, ϵ meet the conditions of Theorem III.C.2, and be efficiently describable and sampleable as above. There is a constant c so that if f is δ-hard for $\mathsf{BPTIME}(t(n))//\log t(n)$, and $\epsilon > t(n)^{-1/c}$, then the family f_B is $(1-\epsilon)$-hard to $(1-\Omega(\delta))$-compute in semi-uniform time $t(n)^{1/c}$.*

The only difference is that the algorithm C', on set B, generates a circuit V_B rather than values v. The advice becomes (A, V_B), and when we generate B' with $A \cup x \subseteq B'$, we use the algorithm to compute the circuit $V_{B'}$, and then estimate consistency by randomly sampling $O((\log 1/\epsilon)/\delta^2)$ elements $a \in A$ and seeing for how many $V_B(a) \neq V_{B'}(a)$.

Theorem III.A.6 is equivalent to the following corollary of Theorem III.D.3.

Corollary III.D.4. *Let $\mathcal{T}$ be the family of random affine subspaces of dimension d of $\mathbb{F}_q^m$, where $d \geq 8$. For some absolute constant c, if f is δ-hard for $\mathsf{BPTIME}(t(n))//\log t(n)$ then the family $f|_B$ for $B \in \mathcal{T}$ is $(1-\epsilon)$-hard to $(1-\Omega(\delta))$-compute in semi-uniform time $t(n)^{1/c}$, for $\epsilon = \max\{q^{-d/16}, t(n)^{-1/c}\}$. Moreover, each $f|_B$ is equivalent to a function on $d \log q$ bit inputs.*

Finally, we observe that Corollary III.D.4 can be used to prove the following derandomized hardness amplification result.

Theorem III.D.5. *Let $\delta > 0, 2^{\sqrt{n}+1} \geq q \geq 2^{\sqrt{n}}$, and let $\mathcal{T}$ be the family of random affine subspaces of dimension $d = 8$ of $\mathbb{F}_q^m$, let $k(n) = O(\sqrt{n}/\delta)$, and let $t(n) \leq 2^{\sqrt{n}}$. For some absolute constant c, if f is δ-hard for $\mathsf{BPTIME}(t(n))//\log t(n)$ then the function $g(B, y_1, \ldots, y_k) = (f|_B)^k(y_1, \ldots, y_k)$ for $B \in \mathcal{S}$ and $y_1, \ldots, y_k \in B$, is $1 - t(n)^{-1/c}$ hard for $\mathsf{BPTIME}(t(n)^{1/c})//(1/c)\log t(n)$. Moreover, g is equivalent to a function on $O(n)$ bits.*

Proof. Assume we have an algorithm that with probability $\epsilon > t(n)^{1/c} > e^{-k(n)\delta/c}$ produces a circuit that ϵ-computes $g = (f|_B)^k$ in time $t'(n) = t(n)^{1/c}$. Then for each of the $\epsilon/2$ B's where the conditional probability of success for the circuit is at least $\epsilon/2$, we can use the list decoder for our Independent code to get a circuit $1-\Omega(\delta)$ computing f_B in time $t'(n)/\text{poly}(\epsilon)$. In other words, the family $f|_B$ has a semi-uniform algorithm that $1-\Omega(\delta)$ computes it with probability $\text{poly}(\epsilon)$. By Theorem III.D.3, f has a semi-uniform time $t'(n)/\text{poly}(\epsilon)$ algorithm that $(1-\delta)$-computes f with $\text{poly}(\epsilon)$ success, a contradiction to the assumed hardness. □

III.E k-XOR code

Here we prove Theorem III.A.7. We first list-decode a code which is a concatenation of our Independent code and the standard Hadamard code.

Let Ind_k be Independent k-wise direct-product code. Let Had_k be the Hadamard code on messages of size k, i.e., for every message $msg \in \{0,1\}^k$, the encoding $Had_k(msg)$ is a function mapping a string $r \in \{0,1\}^k$ to the inner product $\langle msg, r\rangle$ over the binary field $\mathbb{F}_2$. Define $Code_k$ to be the concatenation of Ind_k and Had_k, i.e., $Code_k(f)$ is a function mapping $(x_1,\ldots,x_k;r)$ to $\sum_{i=1}^{k} f(x_i)\cdot r_i \mod 2$, for $x_i \in \{0,1\}^n$ and $r \in \{0,1\}^k$.

We will list-decode this code, using the algorithm of [GL89] for the Hadamard code, and our algorithm $\mathcal{A}$ for the Independent code. First we state the result of Goldreich and Levin.

Theorem III.E.1 ([GL89])**.** *There is a probabilistic algorithm A with the following property. Let $h \in \{0,1\}^k$ be any string , and let $B : \{0,1\}^k \to \{0,1\}$ be any predicate such that $|Pr_{r\in\{0,1\}^n}[B(r) = \langle h, r\rangle] - 1/2| \geq \gamma$, for some $\gamma > 0$. Then, given oracle access to B and given γ, the algorithm A runs in time poly($k, 1/\gamma$), and outputs a list of size $l = O(1/\gamma^2)$ such that with high probability the string h is on this list.*

Using this GL algorithm of Theorem III.E.1, we will show the following.

Theorem III.E.2. *The code $Code_k$ is efficiently, locally, δ-approximately $(1/2+\epsilon, O(1/\epsilon^2))$-list decodable, for $\delta = O(\log 1/\epsilon/k)$.*

Proof. Let C' be the circuit which $(1/2+\epsilon)$-computes $Code_k(f)$. For a given k subset $\bar{x}=(x_1,\ldots,x_k)$, define $\gamma_{\bar{x}} = \mathbf{Pr}_r[\langle f^k(\bar{x}), r\rangle = C'(\bar{x};r)] - 1/2$. Clearly, we have $\mathbf{Exp}_{\bar{x}}[\gamma_{\bar{x}}] \geq \epsilon$ (since C' $(1/2+\epsilon)$-computes $Code_k(f)$).

For a given $\bar{x}=(x_1,\ldots,x_k)$, we set $h=f^k(\bar{x})$ and $B(r)=C'(\bar{x};r)$, and run the GL algorithm with $\gamma=\epsilon/2$. For every $\bar{x}$ with $|\gamma_{\bar{x}}|\geq \epsilon/2$, the GL algorithm will return a list $h_1,\ldots,h_l$ of size $l=O(1/\epsilon^2)$ that, with high probability, contains h.

For each h_i on the list, define $\gamma_{\bar{x},i} = Pr_r[\langle h_i, r\rangle = C'(\bar{x};r)] - 1/2$. By random sampling, we can efficiently estimate each $\gamma_{\bar{x},i}$ to within a constant factor, with high probability. Let $\tilde{\gamma}_{\bar{x},i}$ denote the corresponding approximation. We will choose string h_i with probability proportionate to $(\tilde{\gamma}_{\bar{x},i})^2$, i.e., with probability $(\tilde{\gamma}_{\bar{x},i})^2/\sum_{j=1}^{l}(\tilde{\gamma}_{\bar{x},j})^2$.

For the analysis, first observe that $2\gamma_{\bar{x},i}$ is the discrete Fourier coefficient at h_i of the Boolean function $C'(\bar{x},\cdot)$. By Parseval's identity, we have $\sum_{j=1}^{l} 4\cdot\gamma_{\bar{x},i}^2 \leq 1$. Assuming that we have constant-factor approximations of all $\gamma_{\bar{x},i}$'s and that h was on the list, we conclude that the described algorithm outputs h with probability $\Omega(\gamma_{\bar{x}}^2)$. Since the assumed two events happen with high probability, we get that the probability of producing h is at least $\alpha\cdot\gamma_{\bar{x}}^2$ for some absolute constant $\alpha>0$.

Denote by X the set of all inputs $\bar{x}$, and by G the set of those $\bar{x}$ where $|\gamma_{\bar{x}}|\geq\epsilon/2$. The probability (over a random $\bar{x}$ and internal randomness) that the described algorithm outputs the correct string $f^k(\bar{x})$ is

$$(1/|X|)\sum_{\bar{x}\in G}\alpha\cdot\gamma_{\bar{x}}^2 \geq (1/|X|)\left(\sum_{\bar{x}\in X}\alpha\cdot\gamma_{\bar{x}}^2 - \sum_{\bar{x}\in X\setminus G}\alpha\cdot\gamma_{\bar{x}}^2\right).$$

The first term is α times $Exp_{\bar{x}}[\gamma_{\bar{x}}^2] \geq (Exp_{\bar{x}}[\gamma_{\bar{x}}])^2 \geq \epsilon^2$, by Cauchy-Schwarz and the lower bound $Exp_{\bar{x}}[\gamma_{\bar{x}}]\geq\epsilon$. The second term is at most $\alpha\cdot\epsilon^2/4$ by the definition of G. So the overall success probability of the described algorithm at computing f^k is at least $\Omega(\epsilon^2)$.

Finally, we apply Theorem III.A.3 to the described algorithm for f^k, concluding that the code $Code_k$ is efficiently, locally, δ-approximately $(1/2+$

$\epsilon, O(1/\epsilon^2))$-list decodable, for $\delta = O((\log 1/\epsilon)/k)$. $\square$

To prove Theorem III.A.7, we will show how to list-decode the code obtained by concatenating Ind_{2k} with the truncated Hadamard code $Had_{2k,k}$ where the given $2k$-bit message msg is encoded by the string of inner products $\langle msg, r \rangle$ mod 2, over all $2k$-bit strings r of Hamming weight exactly k. More precisely, we consider the following code $Code(x_1, \ldots, x_{2k}; r) = \sum_{i=1}^{2k} f(x_i) r_i \mod 2$, where $r \in \{0,1\}^{2k}$ have Hamming weight exactly k.

First we observe that given a circuit C which $(1/2 + \epsilon)$-computes the k-XOR encoding of f, the following circuit C' will $(1/2 + \epsilon)$-compute the encoding $Code$ defined above: "Given $(x_1, \ldots, x_{2k}; r)$, for $x_i \in \{0,1\}^n$ and $r \in \{0,1\}^{2k}$ of Hamming weight k, let $y_1, \ldots, y_k$ be the subset of $(x_1, \ldots, x_{2k})$ corresponding to the k positions i where $r_i = 1$. Output the value $C(y_1, \ldots, y_k)$."

Indeed, for uniformly random $2k$-subsets $(x_1, \ldots, x_{2k})$ and a random string $r \in \{0,1\}^{2k}$ conditioned on having Hamming weight exactly k, our circuit C' runs the circuit C on a uniformly random k-subset $(y_1, \ldots, y_k)$, and hence outputs the value $\oplus_{i=1}^{k} f(y_i) = Code_{2k}(f)(x_1, \ldots, x_{2k}; r)$ with probability at least $1/2 + \epsilon$.

We can also get a circuit C'' that $(1/2 + \Omega(\epsilon/\sqrt{k}))$-computes the code $Code_k$ defined earlier: Given $(x_1, \ldots, x_{2k}; r)$, for $x_i \in \{0,1\}^n$ and $r \in \{0,1\}^{2k}$, output a random bit if the Hamming weight of r is not k. Otherwise, let $y_1, \ldots, y_k$ be the subset of $(x_1, \ldots, x_{2k})$ corresponding to the k positions i where $r_i = 1$. Output the value $C(y_1, \ldots, y_k)$." For the analysis, simply observe that a random $2k$-bit string will have Hamming weight k with probability $\Omega(1/\sqrt{k})$. Conditioned on r being of weight k, we get a correct answer with probability $1/2 + \epsilon$; otherwise, we are correct with probability $1/2$.

Applying Theorem III.E.2 to the circuit C'' will yield a list of $O(k/\epsilon^2)$ circuits, one of which $(1 - \delta)$-computes f.

To get the optimal $O(1/\epsilon^2)$ list size, we will *approximately* list-decode the inner, truncated Hadamard code in $Code$. The idea is as follows. We will mimic the proof of Theorem III.E.2 to argue that with probability $\Omega(\epsilon^2)$ over random

$2k$-tuples $(x_1, \ldots, x_{2k})$ and internal randomness, one can produce a $2k$-tuple of bits $(b_1, \ldots, b_{2k})$ such that for all but $O(\delta)$ fraction of indices $i \in [2k]$, we have $f(x_i) = b_i$. Running the approximate list-decoder of Theorem III.A.4, we then get a list of $O(1/\epsilon^2)$ algorithms, one of which $(1-\delta)$-computes f.

We need the following.

Lemma III.E.3. *Let $a = (a_1, \ldots, a_{2k})$ be any $2k$-bit string, and let B be a function mapping $2k$-bits strings r of Hamming weight k to $\{0,1\}$ such that $\mathbf{Pr}_r[\langle a, r\rangle = B(r)] = 1/2 + \eta$, for some unknown η. Suppose we are given $\gamma > 0$ such that $|\eta| \geq \gamma$. Then there is an algorithm that, given γ and oracle access to B, runs in time* $\mathrm{poly}(k, 1/\gamma)$ *and, with probability $\Omega(\eta^2)$, outputs a $2k$-bit string that agrees with a in all but at most δ' fraction of positions, for $\delta' = O((\log 1/\gamma)/k)$.*

Proof. Given B, we define the following randomized algorithm B' mapping $2k$-bit strings to $\{0,1\}$: "On a given r, if the Hamming weight of r is k, output $B(r)$; otherwise, output a random bit."

It is easy to see that $B'(r)$ will agree with the Hadamard encoding $Had_{2k}(a)$ at r for at least $1/2 + \Omega(\eta/\sqrt{k})$ fraction of $2k$-bit strings r. Running the Goldreich-Levin list-decoding algorithm on this B' with the agreement parameter $\Omega(\gamma/\sqrt{k})$, we get with high probability a list of at most $\ell = O(k/\gamma^2)$ strings $h_1, \ldots, h_\ell$ which contains our string a. Next we describe an algorithm for producing a string approximately equal to a, with probability $\Omega(\eta^2)$.

For each $i \in [\ell]$, define $\eta_i = \mathbf{Pr}_r[\langle h_i, r\rangle = B(r)] - 1/2$, where the probability is over $2k$-bit strings r of Hamming weight k. By random sampling, we can estimate each η_i to within a constant factor, with high probability. Let $\tilde{\eta}_i$ denote the respective approximations.

Let us order $|\tilde{\eta}_i|$'s from largest to smallest, and let us discard all those strings h_i where $|\tilde{\eta}_i| < \gamma/2$. For the remaining strings, assume w.l.o.g. that $|\tilde{\eta}_1| \geq \cdots \geq |\tilde{\eta}_{\ell'}|$. We partition the strings h_i's into groups as follows: Let B_1 be the set of all strings of Hamming distance at most δ from h_1; we call h_1 a leader of cluster B_1. Remove all strings B_1 from our list. Let h_j be the first remaining string (according to the order on $\tilde{\eta}_i$'s). Define B_2 to be the set of all remaining strings of Hamming distance at most δ from h_j; here h_j is a leader of B_2. Remove B_2 from the list, and continue until all strings are partitioned into disjoint clusters $B_1, B_2, \ldots, B_t$. For simplicity of notation, assume that the leaders of these clusters are $h_1, h_2, \ldots, h_t$.

Finally, output a leader h_i with probability $\tilde{\eta}_i^2 / \sum_{j=1}^t \tilde{\eta}_j^2$.

For the analysis, we will need the following

Claim III.E.4. $\sum_{i=1}^t \eta_i^2 \leq 1/2$.

Proof. The idea of the proof is the following. The truncated Hadamard code $Had_{2k,k}$ maps any two far-apart messages msg_1 and msg_2 to the codewords $code_1$ and $code_2$ that are almost Hamming distance $1/2$ apart. Switching from the $\{0,1\}$ alphabet to the $\{1,-1\}$ alphabet, the previous statement means that the normalized inner product $\mathbf{Exp}_r[code_1(r) \cdot code_2(r)]$ of the vectors $code_1$ and $code_2$ is close to 0, where the expectation is over $2k$-bit strings of weight k.

Thus the encodings $y_1, \ldots, y_t$ of the leaders $h_1, \ldots, h_t$, respectively, are pairwise almost orthogonal. It is also easy to see that $2\eta_i = \mathbf{Exp}_r[y_i(r) \cdot B(r)]$, and so η_i's are the projections of the vector B onto vector y_i. If the y_i's were pairwise orthogonal, we would get that $B = \sum_{i=1}^t (2\eta_i) \cdot y_i + B^\perp$, where $B^\perp$ is orthogonal to every y_i, for $i \in [t]$. Taking the normalized inner product of B with itself, we would get $\mathbf{Exp}_r[(B(r))^2] = \sum_{i=1}^t (2\eta_i)^2 + \mathbf{Exp}_r[(B^\perp(r))^2]$. Since $(B(r))^2 = 1$ for every r, we conclude that $\sum_{i=1}^t (2\eta_i)^2 \leq 1$.

In reality, the vectors y_i's are pairwise almost orthogonal, and so the calculations will be slightly more complicated, but will follow the same idea. For notational convenience, denote $\alpha_i = 2\eta_i$. Let us write the vector $B = (\sum_{i=1}^t \alpha_i \cdot y_i) + (B - \sum_{i=1}^t \alpha_i \cdot y_i)$. Also for notational convenience in the rest of the proof, let us denote by $\langle B, y_i \rangle$ the normalized inner product $\mathbf{Exp}_r[B(r) \cdot y_i(r)]$. We have

$$\begin{aligned} 1 = \langle B, B \rangle &= \sum_{i,j} \alpha_i \alpha_j \cdot \langle y_i, y_j \rangle + 2\langle \sum_{i=1}^{t} \alpha_i \cdot y_i, B - \sum_{i=1}^{t} \alpha_i \cdot y_i \rangle \\ &\quad + \langle B - \sum_{i=1}^{t} \alpha_i \cdot y_i, B - \sum_{i=1}^{t} \alpha_i \cdot y_i \rangle. \end{aligned}$$

The last term on the right-hand side is nonnegative, and after dropping it we get the following:

$$1 \geq 2 \sum_i \alpha_i^2 - \sum_{i,j} \alpha_i \alpha_j \cdot \langle y_i, y_j \rangle = \sum_i \alpha_i^2 - \sum_{i \neq j} \alpha_i \alpha_j \cdot \langle y_i, y_j \rangle.$$

Hence, $\sum_i \alpha_i^2 \leq 1 + \sum_{i \neq j} \alpha_i \alpha_j \cdot \langle y_i, y_j \rangle$. Since $|\alpha_i| \leq 1$ for all i, the latter is at most $1 + t^2 \cdot \max_{i \neq j}\{\langle y_i, y_j \rangle\}$.

To finish the proof, we need to upperbound t and $\max_{i \neq j}\{\langle y_i, y_j \rangle\}$. We start with the latter. Consider any two $2k$-bit messages msg_1 and msg_2 that differ in at least δ' fraction of positions. Then the normalized inner product of their respective encodings (in the $\{1, -1\}$ alphabet) will be $\mathbf{Exp}_r[(-1)^{\langle msg_1 \oplus msg_2, r \rangle}]$, where r ranges over all $2k$-bits strings of Hamming weight k. Using the Chernoff-Hoeffding bounds, this expectation can be upperbounded by $e^{-\Omega(\delta' k)}$.

The bound on t can be obtained by the Johnson bound: if $y_1, \ldots, y_t$ have pairwise inner products at most $e^{-\Omega(\delta' k)}$ in absolute value, and each $|\langle y_i, B \rangle| \geq \gamma$, for $i \in [t]$, then $t \leq 1/(\gamma^2 - e^{-\Omega(\delta' k)})$ (see,e.g., [IJK06]). For $\delta' = d(\log 1/\gamma)/k$ for a large enough constant d, we get that $t^2 \cdot \max_{i \neq j}\{\langle y_i, y_j \rangle\} \leq 1$. The claim follows. $\square$

Suppose that our string a was put in a cluster with a leader h_i. This means that the (approximation of the) agreement of the truncated Hadamard encoding of h_i with $B(r)$ is at least as big as that of a with $B(r)$ (in absolute values), and that a and h_i are at most δ' Hamming distance apart. We get by the claim above that h_i is output with probability $\Omega(\eta^2)$, as required. $\square$

Proof of Theorem III.A.7. Let $Code$ be the concatenation of Ind_{2k} and the truncated Hadamard $Had_{2k,k}$. As explained earlier, from a circuit C $(1/2+\epsilon)$-computing

$f^{\oplus k}$, we can get C' that $(1/2+\epsilon)$-computes *Code*. For each $2k$-subset $\bar{x} = (x_1, \ldots, x_{2k})$, let $\epsilon_{\bar{x}} = \mathbf{Pr}_r[\langle f^k(\bar{x}), r\rangle = C'(\bar{x}, r)] - 1/2$. Clearly, $\mathbf{Exp}_{\bar{x}}[\epsilon_{\bar{x}}] \geq \epsilon$.

For a given $\bar{x}$, let $a = f^{2k}(\bar{x})$ and let $B(r) = C'(\bar{x}, r)$. Run the algorithm of Lemma III.E.3 on this $B(r)$, with the parameter $\gamma = \epsilon/2$. If $|\epsilon_{\bar{x}}| \geq \gamma$, then we will get with probability $\Omega(\epsilon_{\bar{x}}^2)$ a $2k$-bit string that agrees with a in all but at most δ' positions, for $\delta' = O((\log 1/\epsilon)/k)$.

As in the proof of Theorem III.E.2 above, we then obtain a randomized algorithm for Ind_{2k} that with probability at least $\Omega(\epsilon^2)$ (where the probability is over $\bar{x}$ and the internal randomness of the algorithm) outputs a string that is at most δ' distance away from $f^{2k}(\bar{x})$. Running the approximate list-decoding algorithm for Ind_{2k} from Theorem III.A.4, we obtain a list of $O(1/\epsilon^2)$ circuits, one of which $(1-\delta)$-computes f, for $\delta \leq O(\delta')$. $\square$

III.F Conclusions

We gave an efficient, approximate, local list-decoding algorithm for the direct-product code, with information-theoretically optimal parameters (to within constant factors). Our new decoding algorithm is also very efficient (is in uniform randomized AC^0), and has a simple analysis. We also defined a natural generalization of direct-product codes, intersection codes, for families of subsets $(\mathcal{S}, \mathcal{T})$, and gave the conditions on $(\mathcal{S}, \mathcal{T})$ that suffice for efficient (approximate, local) list-decoding of these generalized codes. Finally, we gave a derandomized version of the direct-product code with an efficient decoding algorithm.

An interesting remaining open question is to get a *derandomized* uniform direct-product theorem with better parameters (pushing the error ϵ to $e^{-\Omega(n)}$, while keeping the new input size linear in the original input size). Another question is to improve the parameters of our approximate version of the uniform direct-product theorem (Theorem III.A.4), ideally achieving a uniform version of the "Chernoff-type" direct-product theorem in the spirit of [IJK07]. Finally, it is interesting to see if the ideas from our new list-decoding algorithm can help in improving the known uniform hardness amplification results for NP of [Tre05].

Acknowledgements : A summary of the results in this Chapter appeared in the paper "Uniform Direct Product Theorems: Simplified, Optimized, and Derandomized" co-authored with Russell Impagliazzo, Valentine Kabanets and Avi Wigderson. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC'08), pages 579–588, 2008.

IV

Uniform Hardness Amplification

IV.A Introduction

In the first chapter we briefly discussed Average-case complexity. In contrast with classical complexity theory, here we are interested in solving problems on the average (that is many instances) rather than in the worst case (all instances). The motivation for this is that worst-case analysis might be too strong a requirement as far as problem solving is concerned. For most practical cases we might be only interested in solving problems on instances which occur frequently. Furthermore, there are classes of problems like NP for which it is unknown if there are efficient algorithms which solve these problems in the worst-case and is even difficult to prove/disprove the same. Analyzing these problems in the average case is one of the ways to make progress on these problems. One question that we re interested in is how the degrees of average case hardness within some complexity class, say NP, are related with respect to some fixed model of computation (e.g. circuits, randomized turing machines). For example, if there is a language within NP which is hard on the average for polynomial size circuits, then is there another language within NP which is even harder for polynomial size circuits? What about the same question with respect to randomized turing machines running in polynomial time? This question is referred to as "uniform hardness amplification within

NP".

In the previous chapters, we have already seen similar hardness amplification results and it would be natural to try to extend these results to resolve some of the above questions. Let us start by setting up the basic definitions. Firstly, language recognition can be interpreted as function computation if we consider the characteristic function of a language L.

Definition IV.A.1 (Characteristic of a Language)**.** Let L be a language. The characteristic of L is a mapping $L : \{0,1\}^* \rightarrow \{0,1\}$ defined as $L(x) = 1$ iff $x \in L$.

Given this, we can talk about hardness of computing $L, L^k, L^{\oplus k}$ on the average. Note that we cannot use our direct product theorems since L^k is not a boolean function but we can use our analogous XOR lemmas. Let us now see some of the problems we might encounter in resolving questions related to hardness amplification within NP.

1. *XOR construction*: We can show that for any NP function L [1], if L is hard to compute on average then $L^{\oplus k}$ is even harder. However we cannot use this for proving hardness amplification results within NP. The simple reason is that for any NP function L, $L \oplus L$ is not an NP function unless $\mathsf{NP} = co\mathsf{NP}$.

   Remember that our proof of the xor lemma was through the direct product theorem. Given that L is hard on the average, we showed that L^k becomes even harder and then extended the hardness result to $L^{\oplus k}$. Instead of using $\oplus$ as the combination function we can use a combination function $g : \{0,1\}^k \rightarrow \{0,1\}$ which ensures that $g(L(.), ..., L(.))$ is also an NP function. Monotone functions which are defined as follows satisfy this property.

   Definition IV.A.2 (Monotone Function)**.** Let $k \in \mathbb{N}$. A function $g : \{0,1\}^k \rightarrow \{0,1\}$ is called monotone if for any pair of strings $x, y \in \{0,1\}^k$, $x \prec y$, $g(x) = 1$ iff $g(y) = 1$. Here $x \prec y$ denotes that $\forall i \in \{1, ..., k\}, x[i] = 1 \Rightarrow y[i] = 1$.

[1] by NP function we mean the characteristic function of a language in NP

So, then the question is can we can extend the hardness result for L^k to $g(L(.), ..., L(.))$ as we did for $L(.) \oplus ... \oplus L(.)$? There also might be an independent way to prove the this than going through the direct product construction.

2. *Uniformity*: Another issue that comes up when using direct product theorems and the xor lemma to show hardness amplification within some complexity class is that of uniformity. In Chapter 2 we saw how a certain minimum amount of nonuniformity was necessary for proving direct product theorems and xor lemma. That is, if we wanted to show that for a function $f : \{0,1\}^* \rightarrow \{0,1\}$, f^k is very hard to compute for probabilistic turing machines, then we have to start with the assumption that f is hard to compute for probabilistic turing machines which take some amount of advice. This poses a problem when we are interested in uniform hardness amplification where we want to show that if there is a language which is hard with respect to BPP algorithms then there is another language in the same complexity class which is even harder with respect to BPP algorithms.

   For certain complexity classes like NP small amount of advice can be removed. More specifically, if there is a language in NP which is hard with respect to BPP algorithms with small amount of advice, then there is another language which is hard with respect to BPP algorithms with no advice. We will discuss these results in the later section of this Chapter.

In a nice paper, Ryan O' Donnell used monotone combination functions to show hardness amplification within NP with respect to polynomial size circuits. The parameters were greatly improved by Healy, Vadhan, and Viola though for languages which are balanced. Below we give the main theorems of both these works.

Theorem IV.A.3 ([O'D04]). *For any polynomial p, if there is a language in NP which is $(1/p(n))$-hard for polynomial size circuit families, then there is another* *language in* NP *which is* $(1/2 - 1/n^{0.33})$*-hard for polynomial size circuit families.*

Theorem IV.A.4 ([HVV04])**.** *For any polynomials* p *and* q*, if there is a* balanced *language*[2] *in* NP *which is* $(1/p(n))$*-hard for polynomial size circuit families, then there is another balanced language in* NP *which is* $(1/2 - 1/q(n))$*-hard for polynomial size circuit families.*

Trevisan proved hardness amplification results with respect to uniform algorithms. The bounds that were obtained were worse compared to the above results and it was mainly due to large amount of nonuniformity required in the proofs of these hardness amplification results. Here is the theorem that was proved.

Theorem IV.A.5 ([Tre05])**.** *For any polynomial* p*, if there is a language in* NP *which is* $(1/p(n))$*-hard for probabilistic polynomial time algorithms, then there is another language in* NP *which is* $(1/2 - 1/(\log n)^{\alpha})$*-hard for probabilistic polynomial time algorithms. Here* $\alpha > 0$ *is a small absolute constant.*

In our study of uniform direct product theorems and uniform XOR lemma, we have obtained the optimal bounds for the nonuniformity required in such arguments. The only drawback when using our results to show hardness amplification within complexity class NP is that we use XOR as the combination function (as opposed to some monotone function). In the next section, we use our results to show hardness amplification within a complexity class where XOR's of multiple languages is still in the same complexity class.

IV.B Uniform hardness amplification in $\mathsf{P}^{\mathsf{NP}_\parallel}$

In this section, we show uniform hardness amplification within the complexity class $\mathsf{P}^{\mathsf{NP}_\parallel}$, the class of problems reducible to NP through one round of parallel oracle queries. More specifically, we prove the following theorem.

[2] *a language such that its characteristic function is balanced for each input size.*

Theorem IV.B.1 (Hardness amplification within $\mathsf{P}^{\mathsf{NP}_{\parallel}}$)**.** *For all sufficiently large n and any constants c, d, if there is a function $f \in \mathsf{P}^{\mathsf{NP}_{\parallel}}$ which is $(1/n^c)$-hard for probabilistic polynomial time algorithms then there is another function $g \in \mathsf{P}^{\mathsf{NP}_{\parallel}}$ which is $(1/2 - 1/n^d)$-hard for probabilistic polynomial time algorithms.*

Our Uniform XOR Lemma, Lemma III.A.2 immediately gives us hardness amplification for probabilistic algorithms with small amount of advice.

Lemma IV.B.2. *Let $f : \{0,1\}^* \to \{0,1\}$ be a Boolean function. For all sufficiently large n and any constants c, d, if f is $1/n^c$-hard for probabilistic polynomial-time algorithms with $O(\log n)$-size (randomness dependent) advice. Then the function $f^{\oplus k}$, for $k = \Omega(n^{2c})$, is $(1/2 - 1/n^d)$-hard for probabilistic polynomial-time algorithms.*

First, we observe that our Lemma III.A.2 immediately gives us hardness amplification in the nonuniform setting with very small amount of advice.

Lemma IV.B.3. *Let $\delta, \epsilon : \mathbb{N} \to [0,1]$ and $f : \{0,1\}^* \to \{0,1\}$ be functions. For all sufficiently large n, if there is a probabilistic polynomial-time algorithm that agrees with the function $f^{\oplus k}$ on at least $1/2 + \epsilon(n)$ fraction of inputs, then there is a probabilistic polynomial-time algorithm that, given advice of size $O(\log 1/\epsilon(n))$, agrees with f on at least $1 - \delta(n)$ fraction of inputs, where $\delta(n) \geq \frac{c \log 1/\epsilon(n)}{k}$ for some constant c.*

Proof. On an input x, we do the following. Given a probabilistic algorithm $(1/2 + \epsilon)$-computing the XOR function $f^{\oplus k}$, we apply to it the algorithm of Lemma III.A.2 from the previous chapter for some poly$(1/\epsilon)$ number times. This gives us a list of poly$(1/\epsilon)$ Boolean circuits such that, with probability exponentially close to 1, at least one of the circuits on the list $(1-\delta)$-computes f. The advice string of size $\log(\text{poly}(1/\epsilon)) = O(\log 1/\epsilon)$ can then be used to identify the correct circuit on the list. We output the result of running this circuit on the input x. □

Now Lemma IV.B.2 is an immediate corollary of Lemma IV.B.3 above.

Logarithmic advice can sometimes be eliminated. For functions in NP, we can use the average-case search-to-decision reduction due to Ben-David, Chor, Goldreich, and Luby [BDCGL92] to obtain the following lemma.

Lemma IV.B.4. *Suppose there is a language $L \in \mathsf{NP}$ and a constant c such that L is $1/n^c$-hard with respect to probabilistic polynomial-time algorithms. Then there is a language $L' \in \mathsf{NP}$ and a constant d such that L' is $1/n^d$-hard with respect to probabilistic polynomial-time algorithm taking $O(\log n)$ bits of advice.*

We will need the average-case "search-to-decision" reduction for NP from [BDCGL92]; we state this result in the form it was stated in [Tre05].

Lemma IV.B.5 ([BDCGL92])**.** *Let $L \in \mathsf{NP}$ be any language, and let $R(\cdot, \cdot)$ be the polynomial-time relation defining L, where $|y| \leq w(|x|)$ for some polynomial function $w(\cdot)$. Then there exist a language $L' \in \mathsf{NP}$, a polynomial $l(\cdot)$, and a probabilistic polynomial-time algorithm A such that the following holds. Given a circuit C' that $(1 - \delta')$-computes L' on inputs of length $l(n)$, the algorithm A outputs, with probability at least $1 - 2^{-\mathrm{poly}(n)}$, a circuit C that solves the search version of L (with respect to R) on at least $1 - \delta$ fraction of inputs of size n, for $\delta = O(\delta' w^2(n))$.*

Proof of Lemma IV.B.4. Let L be a given $\delta = 1/n^c$-hard language in NP, let R be its defining relation, and let $w(n) = n^a$ be the upper-bound on the witness size for n-bit inputs in L. We claim that the language $L' \in \mathsf{NP}$ given in Lemma IV.B.5 is $\delta' = \Omega(\delta/w^2(n))$-hard, on $l(n)$-bit inputs, with respect to probabilistic polynomial-time algorithms with $O(\log n)$ bits of advice.

Indeed, suppose there is an advice-taking probabilistic polynomial-time algorithm that $(1 - \delta')$-computes L'. Enumerate all polynomially many advice strings used by this algorithm, getting a list of polynomially many circuits such that, with probability exponentially close to 1, at least one of the circuits on the list will $(1 - \delta')$-compute L'. Apply the probabilistic polynomial-time algorithm A of Lemma IV.B.5 to each of the circuits on the list. This yields a list of circuits such that, with high probability, at least one of them solves the search version of L (with respect to R) for at least $1-\delta$ fraction of inputs. Simulate each of these circuits on a given input x, and accept iff at least one of them produces a witness y such that $R(x,y)$ holds. It follows that we have a probabilistic polynomial-time algorithm $(1-\delta)$-computing L, contradicting the assumed hardness of L. $\square$

Combining Lemma IV.B.2 and Lemma IV.B.4, we obtain the following.

Theorem IV.B.6. *Suppose there is a Boolean function family $f \in \mathsf{NP}$ and a constant c such that f is $1/n^c$-hard with respect to probabilistic polynomial-time algorithms. Then there is a Boolean function family $g \in \mathsf{P}^{\mathsf{NP}_\parallel}$ that cannot be computed by any probabilistic polynomial-time algorithm on more that $1/2 + 1/n^d$ fraction of inputs, for any constant d.*

Finally, we observe that the existence of a hard function in $\mathsf{P}^{\mathsf{NP}_\parallel}$ implies the existence of a hard function in NP, and so Theorem IV.B.6 can be used to achieve uniform hardness amplification in $\mathsf{P}^{\mathsf{NP}_\parallel}$ claimed in Theorem IV.B.1.

Proof of Theorem IV.B.1. Let f be computed by a SAT-oracle Turing machine M in time n^e, for some constant e. Define a new Boolean function h as follows: For $x \in \{0,1\}^n$ and $i \in [n^e]$, $h(x,i) = 1$ iff M on input x makes at least i SAT-oracle queries and the ith SAT-oracle query is a satisfiable formula.

Clearly, the function h is in NP: Given x and i, we simulate the oracle machine M on x, recording all SAT-oracle queries asked by M — this can be done since M asks all its queries *in parallel*. If the number of queries is less than i, we reject. Otherwise, we nondeterministically check that the ith SAT-oracle query is a satisfiable formula.

Next we argue that the function h is at least $1/(n^c n^e)$-hard with respect to probabilistic polynomial-time algorithms. Indeed, suppose this is not the case. Then there is a probabilistic polynomial-time algorithm A such that for each of at least $1 - 1/n^c$ fraction of xs the algorithm A errs on less than $1/n^e$ fraction of inputs (x,i). Since, for each such x, the number of different inputs (x,i) is at most

n^e, we conclude that A is correct on all SAT-oracle queries made by the machine M on input x. Hence, using this algorithm A to answer SAT-oracle queries, we get a probabilistic polynomial-time algorithm $(1-1/n^c)$-computing f, which contradicts the assumed hardness of f.

Applying Theorem IV.B.6 to the NP-function h, we get the required hard function $g \in \mathsf{P}^{\mathsf{NP}_\parallel}$. $\square$

Trevisan [Tre05] gives uniform hardness amplification for NP: If NP contains a language that is $1/\text{poly}(n)$-hard with respect to probabilistic polynomial-time algorithms, then NP contains a language that is $(1/2 - 1/\log^\alpha n)$-hard, for some constant α. Our Theorem IV.B.1 achieves much better hardness amplification: from $1/\text{poly}(n)$-hardness to $(1/2 - 1/n^d)$-hardness for any d. However, it only applies to the class $\mathsf{P}^{\mathsf{NP}_\parallel}$ rather than NP.

Acknowledgements This Chapter is, in part, based on the paper "Approximately List-decoding Direct Product Codes and Uniform Hardness Amplication" co-authored with Russell Impagliazzo and Valentine Kabanets. In Proceedings of the Forty-Seventh Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), pages 187–196, 2006.

V

Chernoff-type Direct Product Theorems

(This Chapter a joint work with Russell Impagliazzo and Valentine Kabanets.)

V.A Introduction

We studied direct product theorems which were formal statement of the intuition: "if solving one instance of a problem is hard to solve, then solving multiple instances of the problem is even harder". Intuitively, if a problem is hard to solve on the average, then solving more than the expected number of problem instances from a pool of given instances should be very hard. We call such statements "Chernoff-type direct product theorems". We discuss such results in this Chapter. As in the traditional direct product theorems, we start by looking at a simple scenario of boolean functions with fixed input length computed by boolean circuits. Here we will show that if a boolean function f is hard to solve on the average, then not only is the direct product function f^k become harder to compute (previous direct product theorems) but it also becomes harder to approximate f^k (Chernoff-type direct product theorem). We discuss this in the next section.

V.B A Simple Chernoff-type Direct Product Theorem for Boolean Functions

In this section we look at a simple Chernoff-type direct product theorem with respect to boolean functions against boolean circuits. We will see how the proof is a mere generalization of the Trust Halving Strategy that was used to prove traditional direct product theorem in the first Chapter. We will need the following definition for the approximate hardness of direct product of functions.

Definition V.B.1 (Approximate hardness of direct product of functions)**.** Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a boolean function, $k, s \in \mathbb{N}$, and ν, ϵ be constants. We call the direct product function f^k, ν-approximately ϵ-hard for circuits of size s if any circuit C of size at most s, we have

$$\mathbf{Pr}_{(x_1,...,x_k)}[|\{i : C(x_1, ..., x_k)[i] \neq f(x_i)\}| \leq \nu \cdot k] \leq \epsilon$$

Theorem V.B.2 (Chernoff-type Direct Product Theorem for Boolean Functions)**.** *Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a boolean function, $k, s \in \mathbb{N}$, and γ, δ be constants. If f is δ-hard for circuits of size s, then f^k is $(1-\gamma)\delta$-approximately ϵ-hard for circuits of size s', where $\epsilon = (8/\gamma) \cdot e^{-\gamma^2 \delta k/4}$ and $s' = s \cdot poly(\epsilon, \delta, \gamma, 1/n, 1/k)$.*

Proof. The proof is using a generalization of the trust halving strategy that we saw in the first Chapter. For the sake of contradiction let us assume that there is a circuit C such that $\mathbf{Pr}_{(x_1,...,x_k)}[|\{i : C'(x_1, ..., x_k)[i] \neq f(x_i)\}| \leq (1-\gamma)\delta k] > \epsilon$. For any k-tuple $(x_1, ..., x_k)$, let $t(x_1, ..., x_k)$ denote the subset of indices at which C makes an error when computing f^k for $(x_1, ..., x_k)$, that is, $t(x_1, ..., x_k) = \{i : C(x_1, ..., x_k)[i] \neq f(x_i)\}$.

For any fixed subset $H \subseteq \{0,1\}^n$ such that $|H| = 2\delta 2^n$, let $l(x_1, ..., x_k) \overset{def}{=} \{i : x_i \in H\}$. Using Chernoff bounds we know that

$$\mathbf{Pr}_{(x_1,...,x_k)}[|l(x_1, ..., x_k)| \leq (1-\gamma/2)2\delta k] \leq e^{-\gamma^2 \delta k/4} \leq \gamma\epsilon/8$$

Consider the following circuit C' for computing the function f on uniformly chosen inputs in H.

Given a uniformly chosen input $x \in H$, C' chooses $x_1, ..., x_{k-1} \in \{0,1\}^n$ and $i \in \{1, ..., k\}$ uniformly at random. It constructs a k-tuple $(y_1, ..., y_k) \overset{def}{=} (x_1, ..., x_{i-1}, x, x_i, ..., x_{k-1})$. It then queries the circuit C on this k-tuple. If $|t(y_1, ..., y_k)| \leq (1-\gamma)\delta k$, then output $C(y_1, ..., y_k)[i]$, else output $C(y_1, ..., y_k)[i]$ with probability $(1-\gamma/2)^{|t(y_1,...,y_k)|-(1-\gamma)\delta k}$ and with remaining probability it repeats. Finally if no output is produced in $O(n/\epsilon)$ steps, a null answer $\perp$ is produced.

Let us now analyze the advantage of C' over uniform inputs in H. The manner in which an input for the circuit C is constructed induces a distribution over the k-tuples. The probability that a $(y_1, ..., y_k)$ is constructed is precisely $\frac{|l(y_1,...,y_k)|}{2\delta k}$ times the probability of sampling this tuple from the uniform distribution over $\{0,1\}^{nk}$. Given this, we can analyze the advantage of the circuit by evaluating the advantage conditioned on sampling the following subset of tuples:

1. $S_1 = \{(y_1, ..., y_k) : |l(y_1, ..., y_k)| \leq (1-\gamma/2)2\delta k\}$ Since $|S_1|/2^{nk} \leq \gamma\epsilon/8$, the contribution of S_1 to the total advantage is not smaller than $-\gamma\epsilon/8$.

2. $S_2 = \{(y_1, ..., y_k) : |l(y_1, ..., y_k)| > (1-\gamma/2)2\delta k, |t(y_1, ..., y_k)| \leq (1-\gamma)\delta k\}$ The conditional advantage of the C' on these tuples is $\left(1 - \frac{2|t(y_1,...,y_k)|}{|l(y_1,...,y_k)|}\right) \geq \left(1 - \frac{1-\gamma}{1-\gamma/c_2}\right)$. Furthermore, $|S_2| \geq \epsilon - \epsilon/8$ and each tuple in S_2 is sampled with probability $(1-\gamma/2)2^{-nk}$. So the contribution of S_2 in the overall advantage is at least $\gamma(1-1/2)(7\epsilon/8) = 7\gamma\epsilon/16$.

3. $S_3 = \{(y_1, ..., y_k) : |l(y_1, ..., y_k)| > (1-\gamma/2)2\delta k, (1-\gamma)\delta k < |t(y_1, ..., y_k)| \leq (1-\gamma/2)\delta k\}$ The conditional advantage of the C' on these tuples is $\left(1 - \frac{2|t(y_1,...,y_k)|}{|l(y_1,...,y_k)|}\right)$ which is positive. So the total contribution is positive.

4. $S_4 = \{(y_1, ..., y_k) : |l(y_1, ..., y_k)| > (1-\gamma/2)2\delta k, |t(y_1, ..., y_k)| > (1-\gamma/2)\delta k\}$ The conditional advantage of C' is not smaller than $-(1-\gamma/2)^{(1-\gamma/2)\delta k-(1-\gamma)\delta k}$ $\geq -e^{-(1/2)\cdot(1-1/2)\gamma^2\delta k} \geq -\gamma\epsilon/8$.

So the net advantage of the circuit C' over inputs inside H is at least $(7\gamma\epsilon/16 - \gamma\epsilon/8 - \gamma\epsilon/8) = 3\gamma\epsilon/16$.

Now applying the hard-core Theorem I.C.1 from Chapter I we get that there is a circuit C'' of size $|C''| = |C'| \cdot O\left(\frac{1}{\delta^2\gamma^2\epsilon^2}\right)$ which computes f on at least $(1-\delta)$ fraction of the inputs. $\square$

One should note that the same proof argument works if we replace boolean function with a non-boolean function. This is because what we have essentially shown in the above proof is that for any subset of size at least 2δ there is a circuit which achieves a non-negligible advantage over uniformly chosen elements from this subset. The hard-core theorem says that we can "stitch together" such circuits using majorities and argue that for most inputs the correct answers outnumbers the incorrect answers and so we get a circuit which computes f on almost all inputs. The argument breaks down when we try to generalize the results further to multi-valued functions of relations which allows multiple correct answers for the same input.

So, the next question we should ask is whether we can get a Chernoff-type direct product theorem in such scenarios. Such a theorem would be interesting as such scenarios indeed arise in cryptographic settings which we discuss in the next sections.

V.C Cryptographic Setting

Until now in this Dissertation, we have looked at positive and negative results about direct product theorems. We considered problems which were either computing functions or recognizing languages in uniform and nonuniform computational settings. At this point, we would like to generalize and strengthen our results even further and consider such theorems for an even wider range of problem settings. One interesting direction is to consider a setting where solving a problem involves interaction between multiple parties. In classical complexity theory, we have seen how adding such interaction significantly increases the power of computation. So the question for us is in what ways does such interactive settings effect direct product theorems. Are there interactive settings where direct product theorems cannot be proved? In that case can we qualitative/quantitatively find reasons for the same? In the positive direction, what are the circumstances under which we can prove direct product theorems? Such interactive settings occur in cryptography (e.g. Identification) and a positive or negative result about direct product theorems would imply interesting consequences there. So, in this Dissertation we study direct product theorems in interactive settings from the perspective of cryptography. More, specifically we would be interested in challenge-response protocols.

V.C.1 Challenge Response Protocols

These are cryptographic protocols involving two parties, a prover P and a verifier V. We will be interested in the case where both prover and the verifier are efficient, that is, they run in polynomial in some security parameter. Starting with a common input x, the prover and verifier exchange messages in multiple rounds using randomness. Finally, after the end of interaction the verifier either accepts (outputs 1) or rejects (outputs 0). There could be two different kinds of provers, a *cheating prover* and an *honest prover* and the goal is to construct a protocol such that a cheating prover has very small probability of making the verifier accept (where the probability is over the input and randomness involved in the protocol). On the other hand an honest prover should almost always make the verifier accept. The first property is called the soundness while the second property is called completeness.

Such interactive protocols in general are called Interactive arguments and have been studied in the cryptography literature. For instance, consider an identification protocol where a system is trying to distinguish between a human user and a computer program. In this setting the system might generate a random visual challenge (e.g. CAPTCHA) for the user to solve. The assumption is that a computer program would not be able to solve a visual challenge making the protocol sound. The question that we are interested in for such interactive arguments is whether it is possible to amplify the soundness error of a cheating prover by running multiple instances of the protocol in parallel. This is essentially asking if a direct product theorem can be proved in general for interactive protocols.We discuss this issue in the next subsection.

V.C.2 A Cryptographic Scenario where Direct Product Theorem Fails

In this section, we will see a simple protocol where parallel repetition fails to to amplify the soundness error. This shows that we cannot hope to prove a general direct product theorem for interactive arguments. However, we also show that if we restrict ourselves to a subclass of protocols then general direct product theorems can be proved.

More specifically, we will informally discuss a 4-round protocol where parallel repetition fails to amplify the soundness error. However we note that parallel repetition does amplify the soundness error in any 2 or 3 round protocol which was shown in [BIN97] and improved by [CHS05]. We then informally discuss the reasons for this gap. Finally, we start discussing the direct product theorems with respect to $2, 3$ round protocols.

Consider an public key encryption scheme $\mathcal{AE}$ defined by the triple $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. Here $\mathcal{K}$ is the key generation alorithm, $\mathcal{E}$ is the encryption algorithm and $\mathcal{D}$ is the decryption algorithm. $\mathcal{K}$ generates a pair of keys (p_k, s_k), where p_k is called the public key while s_k is called the private key. Given any message M, the encryption of M under the public key is denoted by $C \leftarrow \mathcal{E}_{p_k}(M)$. The encrypted message C can be decrypted given the secret key, that is, $M \leftarrow \mathcal{D}_{s_k}(C)$. For our purposes let us assume that the encryption scheme is perfect. Which means that for any pair of (p_k, s_k) generated by $\mathcal{K}$ and any message M, the encryption of M can be decrypted iff we are given access to the secret key s_k. We will need a stronger property about randomized encryption of a single bit of message known as non-malleability. The property says that for any pair of keys (p_k, s_k) and any bit $b \in \{0,1\}$, given $C_1 \leftarrow \mathcal{E}_{p_x}(b, r_1)$ for any random string r_1, it is not possible to produce a randomized encryption of $(1-b)$, that is, $C_2 \leftarrow \mathcal{E}_{p_k}(1-b, r_2)$ for any random string r_2 than with probability 1/2 unless given s_k. Note that this is a very strong property, since the bit b can be guessed with probability 1/2. Given a non-malleable randomized encryption scheme $\mathcal{AE} = (\mathcal{E}, \mathcal{D}, \mathcal{K})$, consider the following four round protocol between a prover P and verifier V. For $(p_k, s_k) \leftarrow \mathcal{K}$, let p_k be the common input for P and V.

Round 1: V randomly picks a bit b and a random string r and sends $\mathcal{E}_{p_k}(b, r)$.

Round 2: P sends a ciphertext C to V.

Round 3: V sends b and r to P.

Round 4: P sends a bit c and a string s to V.

Finally, V accepts if $C = \mathcal{E}_{p_k}(c, s)$ and $c = 1 - b$. Due to non-malleable property of the ecryption scheme, the verifier has probability 1/2 of succeeding. Now consider a two fold parallel repetition of the above protocol.

Round 1: V randomly picks pair of bits (b_1, b_2) and pair of random strings (r_1, r_2) and sends $\mathcal{E}_{p_k}(b_1, r_1)$, $(\mathcal{E}_{p_k}(b_2, r_2))$.

Round 2: P sends ciphertext (C_1, C_2) to V.

Round 3: V sends (b_1, b_2) and (r_1, r_2) to P.

Round 4: P sends a bit (c_1, c_2) and a string (s_1, s_2) to V.

Finally, V accepts if $\forall i \in \{1,2\}, C_i = \mathcal{E}_{p_k}(c_i, s_i)$ and $c_i = 1 - b_i$. In this case there is a simple strategy for the verifier which succeeds with probability 1/2, thus showing that we cannot get amplification by parallel repetition. The strategy for the prover is to send $(\mathcal{E}_{p_k}(b_2, r_2), \mathcal{E}_{p_k}(b_1, r_1))$ in the second round and send (c_2, c_1), (s_2, s_1) in the fourth round.

As we have seen before, the proofs of the direct product theorems have the following structure: given a challenge or a problem instance, the solver constructs a number challenges for the direct product version of the problem which includes the given challenge and then uses the answers to these parallel challenges to answer the given challenge. This breaks down in four round protocols since challenges and answers in the third and fourth round might be dependent on the challenges and answers of the first two rounds of the parallel version of the protocol. In that case, the prover cannot use the answers of the parallel version of the protocol to construct an answer to single round of challenges.

V.D Chernoff-type Direct Product Theorem for Weakly Verifiable Puzzles

Cryptographic protocols require problems that easy for legitimate users but hard for attackers. The hardness of a problem may be either computational (when attackers are assumed computationally bounded) or information-theoretic (when attackers are computationally unbounded). Ideally, a problem should be reliably easy for legitimate users (i.e., the chance of failure for legitimate users should be negligible), but reliably hard for attackers (i.e., the chance of the attacker's success is negligible). In reality, one may have a problem which is only somewhat easier for legitimate users than for attackers, i.e., the gap between the ability of legitimate users to solve the problem and that of attackers is relatively small. It is thus important to have a method for increasing this gap, thereby improving the security of cryptographic protocols based on such problems.

Direct product theorems provide one such method for making problems reliably hard for attackers. The idea is that if an attacker has some chance of failing on a single challenge, the chance of solving multiple independent challenges should drop exponentially fast with the number of challenges. Examples of such theorems in cryptography include Yao's theorem that weak one-way functions imply strong one-way functions [Yao82] and the results of [BIN97, CHS05] showing similar drops even when an attacker cannot know for certain whether a response to a challenge is correct. Direct product theorems are also important in average-case complexity, circuit complexity, and derandomization. While intuitive, such results are frequently non-trivial to establish. Moreover, there are settings where the intuition is incorrect, and many instances are not proportionally harder; examples where direct products fail are parallel repetition for multiple round protocols and for non-verifiable puzzles [BIN97, CHS05, PW07].

A standard direct product theorem can only be used to amplify the gap between legitimate users and attackers if legitimate users are successful with high probability. Indeed, the legitimate user's chance of solving k independent challenges also drops exponentially fast with k. So unless the legitimate user's probability of failure isn't much more than $1/k$ to start, both legitimate users and attackers will almost certainly fail to solve all of the problems.

Suppose that a legitimate user has probability α of solving a randomly generated challenge, while an attacker has probability $\beta < \alpha$. For k independent random challenges, we expect the legitimate user to be correct on αk of them. By Chernoff bounds, the actual number of correct answers will be very close to αk, with high probability. On the other hand, the expected number of correct answers by the attacker is βk. Intuitively, it should be unlikely that the actual number of correct answers is much larger than the expected number. That is, by analogy with Chernoff bounds, the attacker's probability of answering correctly on significantly more that the expected number βk of random k challenges should be exponentially small in the expectation βk.

This intuition turns out to be correct. The main result of our paper is such a strengthening of the direct product theorem for a very general class of problems, *weakly verifiable puzzles*, introduced in [CHS05].

V.D.1 Example: CAPTCHA

Before defining the class of weakly verifiable puzzles, we consider an example of a cryptographic protocol where our results apply. A CAPTCHA protocol is meant to distinguish between humans and programs, usually using a visual challenge based on distorted text with extraneous lines [ABHL03]. While there seems to be a large gap between the abilities of typical humans and the best current vision algorithms to solve these challenges, algorithms can solve a non-negligible fraction of the puzzles, and many humans (including us) fail a non-negligible fraction of the puzzles.

An obvious, intuitive way to increase the gap is to issue many independent challenges, and accept if the solver is successful on a larger fraction than expected for an attacker, even if the solver does not succeed on all challenges. The fact that *sequential* repetition improves the gap was observed by [ABHL03]. The authors of [ABHL03] also imply that *parallel* repetition improves the gap, referring to the results in [BIN97] for this "more complicated" case. Indeed, the direct product theorem of [BIN97] (improved by [CHS05]) does apply to parallel repetition of CAPTCHA protocols, but it only shows that the probability of algorithmic success decreases with repetitions, not that the gap improves. Our stronger version of the direct product theorem gives the first proof that the parallel repetition protocol suggested in [ABHL03] does indeed improve the gap between legitimate users and attackers.

A CAPTCHA protocol issues a puzzle (e.g., distorted text) such that the correctness of a solution to the puzzle is easy to verify by the generator of the puzzle (who knows the text that was distorted), but not by the attacker (who is just given the puzzle, not the way it was generated). Such puzzles are called *weakly verifiable* in [CHS05].

V.D.2 Weakly verifiable puzzles: Definition and examples

Our result holds for weakly verifiable puzzles defined by [CHS05]. A *weakly verifiable puzzle* has two components:

- a distribution ensemble $D = \{D_n\}_{n\geq 1}$ on pairs (x, α), where x is called the puzzle and α the check string (n is the security parameter); and
- a polynomial-time computable relation $R((x, \alpha), y)$, where y is a string of a fixed polynomially-related length.

The puzzle is thought of as defining a type of challenge x, with y being the solver's response. However, the correctness of the response is not easily verified (and may not be well-defined) given just x. On the other hand, the party generating the puzzle x also knows α, so can verify correctness.

In [CHS05], the distribution D is restricted to be polynomial-time sampleable. In this case, without loss of generality, we can assume that α is the n-bit random tape used to generate the puzzle and check string (if not, we can redefine R as R' which first generate the check string from the random tape, then verifies R). Thus, to simplify the notation in our proofs, we usually assume that α is a uniformly generated n-bit string, and that x is a function of α. A version of our result also holds when D is not polynomial-time sampleable, but only for non-uniform adversaries (since many samples from D are required as advice.)

Here we summarize some important properties of weakly verifiable puzzles. The generation and verification procedures for the puzzles are polynomial-time algorithms. A puzzle may have multiple correct answers (since an answer to a puzzle is verified using a relation). The same puzzle may be generated using multiple random tapes; we call such puzzles *ambiguous*. Moreover, since the verification procedure takes as input the random tape α used to generate a puzzle x, the set of correct answers for x depends on α, and these sets of corrects answers may be different (even disjoint) for different random tapes α and α' that generate the same puzzle x.

Some examples of how weakly verifiable puzzles arise in different settings include:

1. A challenge-response protocol where a prover is trying to get a verifier to accept them as legitimate (e.g., a CAPTCHA protocol where the prover is trying to convince the verifier to accept them as human.) We assume that the verifier is polynomial time with no secret inputs (although an honest prover may have secret inputs). Let α be the random bits used by the verifier. In the first round, the verifier sends a challenge $x = g(\alpha)$, and the prover sends a

response y. The verifier then decides whether to accept by some polynomial time algorithm, $R(\alpha, y)$. Our results are interesting if there is some chance that the honest prover will be rejected, such as an honest human user failing a CAPTCHA challenge based on visual distortion.

2. A secret-agreement protocol with a passive eavesdropper. Let r_A be the random tape used by one party, and r_B that by the other party. Then the conversation C is a function of both r_A, r_B, as is the message m agreed upon. The eavesdropper succeeds if she computes m given C. Then consider $\alpha = (r_A, r_B)$, $x = C$, and $R(C, (r_A, r_B), y)$ if y is the message agreed upon by the two parties using r_A and r_B. Note that there may be some tapes where the parties fail to agree, and thus have no success. Our result shows that, if the parties agree more probably than the eavesdropper can guess the secret, then by running the protocol several times they will almost certainly have more shared secrets than the eavesdropper can guess. Note that, unlike for challenge-response protocols, here there is no restriction on the amount of interaction between the legitimate parties (as long as the eavesdropper is passive).

3. Let f be a (weak) one-way function, and b a (partially-hidden) bit for f, in the sense that it is sometimes hard to always predict b from $x = f(z)$. Since f may not be one-to-one, b may be hard to predict for either information-theoretic or computational reasons. Here, we let $\alpha = z$, $x = f(\alpha)$, and $R(x, \alpha, b')$ if $b' = b(\alpha)$. Our results say that no adversary given an n tuple of $x_i = f(z_i)$ can produce a string closer in relative Hamming distance to $b(\alpha_1) \ldots b(\alpha_n)$ than the hardness of prediction.

4. In the non-uniform setting, our results apply to any function. If f is a function (possibly non-Boolean, or even multi-valued, as long as it takes on at most a polynomial number of values), we can define α to be (the set of all elements in) $f(x)$. Then $y \in f(x)$ if and only if $y \in \alpha$, so this is testable in

polynomial-time given α. This distribution isn't necessarily polynomial-time sampleable, so our results would only apply for non-uniform adversaries (e.g., Boolean circuits.)

Note that in some examples, success may be ill-defined, in that x may not uniquely determine α, and so it may not be information-theoretically possible to know whether $R((x, \alpha), y)$ given only x.

V.D.3 Our main result

Before stating our main theorem, we need several definitions. For a weakly verifiable puzzle P and a natural number k, we denote by P^k the k-wise direct product of P, i.e., P^k is the puzzle that asks k independent challenges from P [1]. For a parameter $0 \leq \delta \leq 1$, we say that P is *δ-hard* for time t if every randomized algorithm running in time $t(n)$ has probability at least δ of answering incorrectly a randomly generated challenge from P (where the probability is both over input challenges from P and the internal randomness of the algorithm), for sufficiently large input size n. Finally, for parameters $k \in \mathbb{N}$ and $0 \leq \nu, \delta \leq 1$, we say that the k-wise direct product puzzle P^k is *ν-approximately δ-hard* for time t if every randomized algorithm running in time $t(n)$ has probability at least δ (over k-tuples of challenges from P^k, and its internal randomness) of answering incorrectly at least νk of the input k challenges.

Our main theorem states that for δ-hard puzzle P, its k-wise direct product P^k is, essentially, δ-approximately $(1 - o(1))$-hard. That is, not only is it impossible to solve all k challenges for a non-negligible fraction of k-tuples from P^k, but also it is impossible to make significantly fewer than the expected number δk of mistakes on the input k challenges. More precisely, we have the following.

Theorem V.D.1 (Main Theorem)**.** *Let P be a weakly verifiable puzzle that is δ-hard for time t. Let $k \in \mathbb{N}$ and $\gamma > 0$ be arbitrary, and let $\epsilon \geq (100/\gamma\delta) \cdot e^{-\gamma^2 \delta k/40}$.*

[1] we consider distribution ensemble D^k on pairs $((x_1, ..., x_k), (\alpha_1, ..., \alpha_k))$, where (x_i, α_i)'s are chosen independently from D. Given that $(y_1, ..., y_k)$ is an answer to this "direct product" version of the puzzle P, P^k is verified by using the relation R individually on the puzzles, i.e., checking $R((x_i, \alpha_i), y_i)$.

Then the direct product puzzle P^k *is* $(1-\gamma)\delta$*-approximately* $(1-\epsilon)$*-hard for time* $t' = t(n) \cdot \mathrm{poly}(\epsilon, 1/n, 1/k)$.

We call this a *Chernoff-type* direct product theorem, since it shows that the "tail bound" on the number of correctly solved puzzles drops exponentially in the region beyond its expectation.

Standard Chernoff bounds show that, if the legitimate user can solve the problem with probability of failure less than, say, $(1-2\gamma)\delta$, then they will succeed in solving all but $(1-\gamma)\delta k$ of the input k challenges, for almost all k-tuples from P^k. Thus our Chernoff-type direct product theorem indeed provides a way to amplify any gap between legitimate users and attackers.

Finally, we should also note that for direct products with threshold it is impossible to get the bound $\epsilon = (1-\delta)^k$, which is possible for standard direct products [CHS05]. Indeed, consider the case of a puzzle P such that P is easy for $(1-\delta)$ fraction of inputs, but is information-theoretically impossible to solve on the remaining δ fraction of inputs. Then the probability of making fewer than δk mistakes on a given random k-tuple of challenges is the tail bound for the binomial distribution where one flips k independent coins with the "heads" probability δ. When δk is sufficiently far from 0 and far from k (e.g., for constant $0 < \delta < 1$), then the Chernoff bound provides a tight estimate for this tail bound. Thus the bound of our main theorem cannot be significantly improved, except possibly for making the constant in the exponent of ϵ in Theorem V.D.1 (currently 40) closer to that of the Chernoff bound (which can be as low as 2).

V.D.4 Related work

The notion of a direct product theorem, where solving multiple instances of a problem simultaneously is proven harder than a single instance, was introduced by Yao in [Yao82]. Due to its wide applicability in cryptography and computational complexity, a number of different versions and proofs of such theorems can be found in the literature; see, e.g., [GNW95] for a good compilation of such results.

In this paper, we use some of the proof techniques (namely the *trust halving strategy*) introduced by Impagliazzo and Wigderson in [IW97]. Such techniques were also used to prove a version of the direct product theorem in a more general cryptographic setting by Bellare, Impagliazzo and Naor in [BIN97]. It is shown in [BIN97] that the soundness error decreases exponentially with parallel repetition in any 3-round challenge-response protocol, but such error amplification might not be possible for a general (> 3)-round protocol. Pietrzak and Wikstrom in [PW07] extend this negative result. On the positive side, Canetti, Halevi and Steiner in [CHS05] used ideas from [BIN97] to define a general class of *weakly verifiable puzzles* for which they show parallel repetition amplifies hardness, also giving a quantitative improvement over [BIN97]. More recently, Pass and Venkitasubramaniam [PV07] show similar positive results for constant round public coin protocols.

All the previous results mentioned above consider parallel repetition without threshold, i.e., they consider the hardness of answering *all* the instances of the parallel repetition question simultaneously.

Comparing the techniques of [BIN97] and those of [CHS05]. Our construction uses a version of the trust-reducing strategy from [IW97, BIN97]. In a trust-reducing strategy, the input puzzle is hidden among $(k-1)$ randomly generated puzzles, and the number of mistakes the attacker makes on the random puzzles is used to compute the probability with which the algorithm trusts the attacker's answer for the input puzzle.

A different approach was used in [CHS05]. Their proof strategy (similar to that of Goldreich, Nisan and Wigderson [GNW95]) is roughly as follows. Suppose that some attacker $\bar{C}$ correctly answers all k challenges for at least ϵ fraction of k-tuples from some direct product puzzle P^k, where P is δ-hard. Then (arguing by induction) one shows that there exists a position $1 \leq i \leq k$ and fixed inputs $x_1, \ldots, x_{i-1}$ for the positions before i such that the probability of getting a correct answer for the ith input in a given k-tuple, conditioned on the attacker's answers for the positions $i+1,\ldots,k$ being correct, is at least $1-\delta$. Thus, to answer a challenge x, one places x into position i, randomly generates challenges for the positions higher than i, runs the attacker $\bar{C}$ on the constructed k-tuple, and outputs the answer of $\bar{C}$ for x if the answers of $\bar{C}$ on *all* the $k-i$ random challenges are correct; otherwise one repeats with new $k-i$ random challenges.

The argument of [CHS05] allows one to conclude that $\epsilon = (1-\delta)^k$, which is information-theoretically the best possible bound, and is a quantitative improvement on the bound on ϵ shown in [BIN97]. While the techniques of [CHS05] yield stronger (optimal) bounds for the direct product than those of [BIN97], we do not see how to use the techniques of [CHS05] for the case of direct products *with threshold* that we consider in the present paper. In our case, we need to deal with a variable number of mistakes even for "good" k-tuples, and we manage to adapt the techniques of [BIN97] to handle such mistakes.

V.D.5 Our techniques

As in [IW97, BIN97], our proof of the main theorem is constructive: we show how to use a breaking strategy that solves the threshold puzzle with probability ϵ as a subroutine in an algorithm that solves a single puzzle with probability greater than $1-\delta$. However, we need to deviate substantially from the previous analysis.

The way it is argued in [IW97, BIN97] that all but δ fraction of inputs are easy is as follows. Suppose an algorithm A succeeds on a significant fraction of k-tuples of random puzzle instances. Then one constructs another algorithm A' such that, for every subset of puzzle instances H of density at least δ, algorithm A' succeeds almost surely on a random instance in H. Now consider the set of all puzzle instances where A' gives a wrong answer. By the above, this set must have density less than δ (or else A' would succeed almost surely on a random element in the set).

In contrast, in the threshold scheme, it is not possible to construct an algorithm A' with the similar guarantee that A' succeeds almost surely on a random instance in H, for every subset H of density at least δ. Indeed, for a given puzzle P, there may be a subset H' of density $(1-\gamma)\delta$ of input instances where no algorithm can succeed with non-negligible probability, while all the other instances outside H' are easy to solve. In this case, there is an algorithm that solves almost all k-tuples of puzzle instances, if we allow up to about $(1-\gamma)\delta k$ errors. However, for any set H of density δ such that $H' \subseteq H$, no algorithm A' can succeed on more than γ fraction of elements of H.

In order to get around this obstacle, we need a more *global* way of analyzing the trust-reducing strategy. Our main tools for doing this are sampling lemmas from [IJKW08]. The high-level idea is as follows. Let G be the set of k-tuples of puzzle instances where some algorithm A is correct in all but $(1-\gamma)\delta k$ positions. Suppose that G has density ϵ. The trust-reducing strategy essentially allows us to construct an efficient oracle for testing membership in G. The overall strategy for solving a puzzle instance x is then to sample random k-tuples containing x, until getting the tuple that falls into G; for such a tuple, we output the value A gives for the xth position in the tuple.

Since G has density ϵ, we are almost sure to sample a tuple from G within poly$(1/\epsilon)$ iterations. We use a sampling lemma to argue that, conditioned on sampling a random k-tuple from G, the position of the input x is distributed almost uniformly within the tuple. Hence, in that case, we get the correct answer for x with probability at least $1-(1-\gamma)\delta = 1-\delta+\gamma\delta$ (since every tuple in G has at most $(1-\gamma)\delta k$ bad positions). Accounting for possible errors of our membership oracle for G, the probability of our sampling procedure missing the set G, and the fact that the xth positions is only almost uniform within the tuple, we conclude that our algorithm succeeds on at least $1-\delta$ fraction of inputs x.

V.E Preliminaries

V.E.1 Basics: Hoeffding bound

For a natural number k, we will denote by $[k]$ the set $\{1, \ldots, k\}$.

Lemma V.E.1 (Hoeffding bound)**.** *Let $X_1, \ldots, X_t$ be independent identically distributed random variables taking values in the interval $[0,1]$, with expectation μ. Let $\chi = (1/t)\sum_{i=1}^{t} X_i$. For any $0 < \nu \le 1$, we have $\mathbf{Pr}[\chi < (1-\nu)\mu] < e^{-\nu^2 \mu t/2}$.*

V.E.2 Samplers

We will consider bipartite graphs $G = G(L \cup R, E)$ defined on a bipartition $L \cup R$ of vertices; we think of L as left vertices, and R as right vertices of the graph G. We allow graphs with multiple edges. For a vertex v of G, we denote by $N_G(v)$ the multiset of its neighbors in G; if the graph G is clear from the context, we will drop the subscript and simply write $N(v)$. Also, for a vertex x of G, we denote by E_x the set of all edges in G that are incident to x. We say that G is *bi-regular* if the degrees of vertices in L are the same, and the degrees of vertices in R are the same.

Let $G = G(L \cup R, E)$ be any bi-regular bipartite graph. For a function $\lambda : [0,1] \times [0,1] \to [0,1]$, we say that G is a λ*-sampler* if, for every function $F : L \to [0,1]$ with the average value $\mathbf{Exp}_{x \in L}[F(x)] \ge \mu$ and any $0 < \nu < 1$, there are at most $\lambda(\mu, \nu) \cdot |R|$ vertices $r \in R$ where $\mathbf{Exp}_{y \in N(r)}[F(y)] \le (1-\nu)\mu$.

We will use the following properties of samplers (proved in [IJKW08] for the special case of $\nu = 1/2$); for completeness, we state them with the proofs. The first property says that for any two large vertex subsets W and F of a sampler, the fraction of edges between W and F is close to the product of the densities of W and F.

Lemma V.E.2 ([IJKW08])**.** *Suppose $G = G(L \cup R, E)$ is a λ-sampler. Let $W \subseteq R$ be any set of measure at least τ, and let $V \subseteq L$ be any set of measure at least β. Then, for all $0 < \nu, \beta < 1$ and $\lambda_0 = \lambda(\beta, \nu)$, we have $\mathbf{Pr}_{x \in L, y \in N(x)}[x \in V \ \&\ y \in$*

$W] \geq \beta(1-\nu)(\tau-\lambda_0)$, where the probability is for the random experiment of first picking a random node $x \in L$ uniformly at random, and then picking a uniformly random neighbor y of x in the graph G.

Proof. We need to estimate the probability of picking an edge between V and W in a random experiment where we first choose a random $x \in L$ and then its random neighbor y. Since the graph G is assumed to be bi-regular, this probability remains the same in the experiment where we first pick a random $y \in R$ and its random neighbor $x \in N(y)$. The latter is easy to estimate using the sampling property of the graph G, as follows. Consider the function $F : L \to [0,1]$ defined as

$$F(x) = \begin{cases} 1 & \text{if } x \in V; \\ 0 & \text{otherwise.} \end{cases}$$

Since the measure of V is at least β, we have $\mathbf{Exp}_{x\in L}[F(x)] \geq \beta$. Let $W' \subseteq W$ be the subset of vertices that have at least $(1-\nu)\beta$ fraction of their neighbors in V. In other words, W' contains those vertices $r \in R$ such that $\mathbf{Exp}_{y\in N(r)}[F(y)] \geq (1-\nu)\beta$. Since G is a λ-sampler and W is of measure at least τ, we get that W' is of measure at least $\tau - \lambda_0$. Then, conditioned on picking a vertex $y \in W'$, the probability that its random neighbor is in V is at least $(1-\nu)\beta$. The lemma follows. □

The second property deals with edge-colored samplers. It basically says that removing some subset of right vertices of a sampler yields a graph which (although not necessarily bi-regular) still has the following property: Picking a random left node and then picking its random neighbor induces roughly the same distribution on the edges as picking a random right node and then its random neighbor.

Lemma V.E.3 ([IJKW08]). *Suppose $G = G(L \cup R, E)$ is a λ-sampler, with the right degree D. Let $W \subseteq R$ be any subset of density at least τ, and let $G' = G(L \cup W, E')$ be the induced subgraph of G (obtained after removing all vertices in*

$R \setminus W$), with the edge set E'. Let $Col : E' \to \{red, green\}$ be any coloring of the edges of G' such that at most $\eta D|W|$ edges are colored red, for some $0 \leq \eta \leq 1$. Then, for all $0 < \nu, \beta < 1$ and $\lambda_0 = \lambda(\beta, \nu)$, we have

$$\mathbf{Pr}_{x \in L, y \in N_{G'}(x)}[Col(\{x, y\}) = red] \leq \max\{\eta/((1-\nu)(1-\lambda_0/\tau)), \beta\},$$

where the probability is for the random experiment of first picking a uniformly random node $x \in L$, and then picking a uniformly random neighbor y of x in the graph G'.

Proof. For every $x \in L$, let d_x be the degree of x in G', and let $\xi(x)$ be the fraction of red edges incident to x in G'. The probability we want to estimate is exactly $\mu = \mathbf{Exp}_{x \in L}[\xi(x)]$. If $\mu \leq \beta$, then we are done. So for the rest of the proof, we will suppose that $\mu > \beta$.

Let $W' \subseteq W$ be the subset of those vertices w where $\mathbf{Exp}_{x \in N(w)}[\xi(x)] \geq (1-\nu)\mu$. (Here we use $N(w)$ to denote the neighborhood $N_{G'}(w)$ of w in G', which is the same as $N_G(w)$ by the definition of G'.) Since G is a λ-sampler and W has measure at least τ in R, we get that W' has measure at least $1 - \lambda_0/\tau$ in W. Hence, we have

$$\sum_{y \in W} \mathbf{Exp}_{x \in N(y)}[\xi(x)] \geq \sum_{y \in W'} \mathbf{Exp}_{x \in N(y)}[\xi(x)] \geq |W|(1-\lambda_0/\tau)(1-\nu)\mu. \quad \text{(V.1)}$$

On the other hand, $\sum_{y \in W} \left(D \cdot \mathbf{Exp}_{x \in N(y)}[\xi(x)]\right)$ is simply the summation over all edges (x, y) in G' where each edge (x, y) with $x \in L$ contributes $\xi(x)$ to the sum. Since the degree of each x is d_x, each $x \in L$ contributes exactly $d_x\xi(x)$, which is the number of incident red edges at x. Hence, the total sum is exactly the number of red edges in G', which is at most $\eta D|W|$ by assumption. It follows that

$$\sum_{y \in W} \mathbf{Exp}_{x \in N(y)}[\xi(x)] = (1/D) \sum_{x \in L} d_x\xi(x) \leq |W|\eta. \quad \text{(V.2)}$$

Finally, comparing the bounds in (V.1) and (V.2), we conclude that $\mu \leq \eta/((1-\nu)(1-\lambda_0/\tau))$. □

We shall also need a generalization of Lemma V.E.3 for the case of weighted graphs. Here we consider bipartite graphs $G = G(L \cup R, E)$ whose edges are assigned weights in the interval $[0, 1]$ satisfying the following property: for every right vertex $y \in R$, all the edges incident on y are assigned the same weight. Let $w_0, w_1, \ldots$ be the distinct weights of the edges of G, in decreasing order. The vertex set R of such a weighted graph is naturally partitioned into subsets $W_0, W_1, \ldots,$ where W_i is the subset of all those vertices in R whose incident edges have weight w_i. Intuitively, such a partitioning of R defines a new induced graph G' where a vertex in W_i is present in G' with probability w_i. (In the setting of Lemma V.E.3, there are two sets $W_0 = W$ and $W_1 = R \setminus W$, with $w_0 = 1$ and $w_1 = 0$.)

Suppose the edges of G are partitioned into red and green edges. Let Red denoted the set of all red edges, and, for every $x \in L$, let Red_x denote the set of all red edges incident to x.

First, consider the experiment where one picks a vertex $y \in R$ with probability proportinate to w_i, where $y \in W_i$, and then picks a uniformly random edge incident to y. What is the probability of picking a red edge?

Let $wt : E \to [0, 1]$ be the edge weight function for our graph $G = G(L \cup R, E)$, and let D be the right degree of the graph G. For a fixed red edge e of G, the probability of choosing this edge in the random experiment described above is

$$\frac{wt(e)}{\sum_{i \geq 0} |W_i| w_i} \cdot \frac{1}{D},$$

where $wt(e)/(\sum_{i \geq 0} |W_i| w_i)$ is the probability of choosing the vertex $y \in R$ that is the end vertex of the edge e, and $1/D$ is the probability of picking one of the D edges incident to y. The probability of picking some red edge is then simply the sum of the probabilities of picking an edge e over all red edges e of G.

Next consider the following experiment. Pick a vertex $x \in L$ uniformly at random, then pick an edge e incident to x with probability proportinate to $wt(e)$ (i.e., the probability $wt(e)/(\sum_{e' \in E_x} wt(e')))$. The probability $\xi(x)$ of picking a red edge incident to x is then the sum of the probabilities of choosing an edge e incident to x, over all red edges e incident on x. Finally, the overall probability of picking a red edge in this experiment is simply the average $\mathbf{Exp}_{x\in L}[\xi(x)]$.

The next lemma basically says that, for sampler graphs G, the probabilities of picking a red edge in the two experiments described above are almost the same. More precisely, we have the following.

Lemma V.E.4. *Suppose $G = G(L \cup R, E)$ is a λ-sampler with the right degree D. Let $wt : E \to [0, 1]$ be the weight function over the edges of G such that, for each $y \in R$, the weights of the edges $e \in E_y$ incident to y are the same. Let $w_0, w_1, \dots$ be the distinct weights of the edges of G, in decreasing order, and let $W_0, W_1, \dots$ be the partitioning of the vertex set R so that each W_i is the subset of all those vertices in R whose incident edges have the weight w_i. Suppose that W_0 has the measure at least τ in the set R.*

Let $Col : E \to \{red, green\}$ be any coloring of the edges of G. For each $x \in L$, let Red_x be the set of all red edges incident to x, and let Red be the set of all red edges in G. Suppose that the total weight of red edges $\sum_{e\in Red} wt(e)$ is at most $\eta D|R|$, and let $\xi(x) = (\sum_{e\in Red_x} wt(e))/(\sum_{e\in E_x} wt(e))$.

Then, for any $0 < \nu < 1$ and $\lambda_0 = \lambda(\beta, \nu)$, we have

$$\mathbf{Exp}_{x\in L}[\xi(x)] \le \max\left\{\frac{\eta|R|}{(1-\nu)(1-\lambda_0/\tau)\sum_{i\ge 0}|W_i|w_i}, \beta\right\}.$$

Proof. Let $\mu = \mathbf{Exp}_{x\in L}[\xi(x)]$. If $\mu \le \beta$, then we are done. So for the rest of the proof, we will suppose that $\mu > \beta$. We will bound the following sum from below, and from above:

$$\sum_{i\ge 0}\sum_{y\in W_i} w_i \cdot \mathbf{Exp}_{x\in N(y)}[\xi(x)]. \qquad \text{(V.3)}$$

To bound it from below, let $Bad \subseteq R$ be the subset of those vertices u where $\mathbf{Exp}_{x\in N(u)}[\xi(x)] < (1-\nu)\mu$. Since G is a λ-sampler, we get that Bad has measure at most λ_0 in R. Each vertex y outside the set Bad contributes at least $w_i(1-\nu)\mu$ to the sum (V.3) for $y \in W_i$. Since $w_0 \ge w_1 \ge \dots$, the sum of such contributions is minimized when all the bad vertices are in W_0, i.e., $Bad \subseteq W_0$ (otherwise, we can always make the sum smaller by placing a bad vertex into W_0 and creating a good vertex in some W_i for $i > 0$). Since W_0 has measure at least τ, we get that Bad has measure at most λ_0/τ in W_0, and so

$$\begin{aligned}\sum_{i\geq 0}\sum_{y\in W_i} w_i \cdot \mathbf{Exp}_{x\in N(y)}[\xi(x)] &\geq (1-\lambda_0/\tau)|W_0|w_0(1-\nu)\mu + \sum_{i\geq 1}|W_i|w_i(1-\nu)\mu \\ &\geq (1-\lambda_0/\tau)(1-\nu)\mu\sum_{i\geq 0}|W_i|w_i.\end{aligned}$$

For the upper bound on the sum in (V.3), we use the definition of the $\xi(x)$, change the order of summation, and finally use the definition of η to obtain the following:

$$\begin{aligned}\sum_{i\geq 0}\sum_{y\in W_i} w_i \cdot \mathbf{Exp}_{x\in N(y)}[\xi(x)] &= (1/D)\sum_{y\in R}\sum_{x\in N(y)} \xi(x)wt((x,y)) \\ &= (1/D)\sum_{x\in L}\xi(x)\sum_{e\in E_x} wt(e) \\ &= (1/D)\sum_{x\in L}\sum_{e\in Red_x} wt(e) \\ &= (1/D)\sum_{e\in Red} wt(e) \\ &\leq |R|\eta.\end{aligned}$$

Comparing the obtained lower and upper bounds, we conclude that

$$\mu \leq \frac{|R|\eta}{(1-\nu)(1-\lambda_0/\tau)\sum_{i\geq 0}|W_i|w_i},$$

completing the proof of the lemma. □

We conclude this section by showing that the direct product gives rise to a sampler. Consider the following bipartite graph $G = G(L\cup R, E)$: the set of left vertices L is the set of n-bit strings $\{0,1\}^n$; the right vertices R are all k-tuples of n-bit strings $\{0,1\}^{nk}$; for every $y = (u_1, \ldots, u_k) \in R$, there are k edges $(y, u_1), \ldots, (y, u_k) \in E$.

Lemma V.E.5. *The graph G defined above is a λ-sampler for $\lambda(\mu, \nu) = e^{-\nu^2 \mu k/2}$.*

Proof. This is immediate from Lemma V.E.1. □

V.F Proof of the Main Theorem

The proof is by contradiction. Suppose a solver $\bar{C}$ solves the direct product puzzle P^k with fewer than $(1-\gamma)\delta k$ mistakes for more than ϵ fraction of k-tuples of puzzle instances. We will describe a solver $\mathcal{C}$ which solves the puzzle P with probability at least $1-\delta$, where the probability is over the internal randomness of the solver and uniformly chosen $\alpha \in \{0,1\}^n$.

We first give the proof under the simplifying assumptions that all puzzles are non-ambiguous (i.e., a puzzle x uniquely determines the random tape α that generated x), and that we can test if a given k-tuple of puzzle instances $(x_1, \ldots, x_k)$ is such that $\bar{C}(x_1, \ldots, x_k)$ makes fewer than $(1-\gamma)\delta k$ mistakes. Later we remove these assumptions.

V.F.1 Proof under simplifying assumptions

Let *Good* be the subset of k-tuples of puzzle instances where $\bar{C}$ makes few mistakes. More precisely, *Good* is the set of those k-tuples of random tapes $(\alpha_1, \ldots, \alpha_k)$ such that, for the corresponding k-tuple of puzzles $(x_1, \ldots, x_k)$, the solver $\bar{C}$ makes fewer than $(1-\gamma)\delta k$ mistakes. Since we assume that all puzzles are non-ambiguous, we can define the set *Good*$'$ of all those k-tuples of puzzles $(x_1, \ldots, x_k)$ such that the k-tuple of corresponding random tapes $(\alpha_1, \ldots, \alpha_k)$ is in G. That is, *Good*$'$ is the set of all k-tuples of puzzle instances where the solver $\bar{C}$ makes fewer than $(1-\gamma)\delta k$ mistakes.[2] Our second assumption is that we have an oracle for testing membership in the set *Good*$'$.

Consider the following algorithm $\mathcal{C}$:

[2]Note that the set *Good*$'$ does not make sense if one allows ambiguous puzzles, as the same instance x may be considered solved correctly or incorrectly by $\bar{C}$ depending on the particular random tape α used to generate that x.

> "On input x, choose $k-1$ random tapes $\alpha_1, \ldots, \alpha_{k-1}$ uniformly at random. Let $x_1, \ldots, x_{k-1}$ be the puzzles corresponding to the chosen random tapes. Pick $i \in [k]$ at random, and set $\bar{x} = (x_1, \ldots, x_{i-1}, x, x_i, \ldots, x_{k-1})$. Test if $\bar{x} \in \mathit{Good}'$ (using the assumed membership oracle for *Good′*). If $\bar{x} \in \mathit{Good}'$, then output $\bar{C}(\bar{x})_i$; otherwise repeat with new random α's and i. If no output is produced within $4\ln(20/\gamma\delta)/\epsilon$ iterations, then output the error symbol $\perp$."

We want to analyze the success probability of solver $\mathcal{C}$ on a given input x. To this end, we need to argue that (1) the probability of the timeout is small, and (2) conditioned on the output being different from $\perp$, it is a correct output with high probability (greater than $1-\delta$).

Recall the λ-sampler G defined at the end of Section V.E.2. It has as its left vertices all possible n-bit random tapes α, and as its right vertices all possible k-tuples of such tapes. For a left vertex α, its neighbors in G correspond to all possible ways of embedding this α in a k-tuple, as done by our algorithm $\mathcal{C}$. The algorithm $\mathcal{C}$ times out on an input x corresponding to the random tape α iff it never samples a neighbor w of α in G such that $w \in \mathit{Good}$. To bound the probability of timeout, we consider the set $H \subseteq \{0,1\}^n$ of all those left vertices α of G such that α has less than $\epsilon/4$ fraction of its neighbors fall into the set *Good*.

Claim V.F.1. *The set H has density at most $\gamma\delta/5$.*

Proof. Suppose that the density of H is greater than $\beta = \gamma\delta/5$. Let $H' \subseteq H$ be any subset of H of density exactly β. By our assumption, we have that $\mathbf{Pr}_{\alpha\in L, w\in N(\alpha)}[\alpha \in H' \ \&\ w \in \mathit{Good}] < \beta\epsilon/4$. On the other hand, by Lemma V.E.2 we get that the same probability is at least $\beta(\epsilon - \lambda_0)/3$ for $\lambda_0 = \lambda(\beta, 2/3)$. This is a contradiction since $\lambda_0 \le \epsilon/4$ from the assumption of the theorem. □

Claim V.F.2. *For every $\alpha \notin H$ and the puzzle x corresponding to that random tape α, we have $\mathbf{Pr}[\mathcal{C}(x) = \perp] \le \gamma\delta/20$, where the probability is over the internal randomness of $\mathcal{C}$.*

Proof. By the definition of H, we get that the probability of timeout on any given $\alpha \notin H$ is at most $(1-\epsilon/4)^{4\ln(20/\gamma\delta)/\epsilon} \le \gamma\delta/20$. □

Next we bound the probability of $\mathcal{C}$ making a mistake, conditioned on $\mathcal{C}$ outputting something other than $\perp$. First we observe that $\mathcal{C}$ produces a definite answer on an input x corresponding to the random tape α exactly when it samples a neighbor w of α in the graph G such that $w \in \mathit{Good}$. Consider the subgraph G' of G induced by removing all right vertices of G except those in Good. For each edge in G' between $w = (\alpha_1, \ldots, \alpha_k) \in R$ and $\alpha_i \in L$, color this edge red if $\bar{C}(x_1, \ldots, x_k)_i$ is wrong, and color it green otherwise. Then the requisite conditional probability of $\mathcal{C}$ making a mistake is exactly $\mathbf{Pr}_{\alpha \in L, w \in N_{G'}(\alpha)}[(\alpha, w) \text{ is red}]$.

Claim V.F.3. $\mathbf{Pr}_{\alpha \in \{0,1\}^n}[\mathcal{C}(x)$ *is wrong* $\mid \mathcal{C}(x) \neq \perp] \leq \delta - \gamma\delta/4$, *where x is the puzzle corresponding to the random tape α.*

Proof. By the discussion above, the required conditional probability is exactly $\mathbf{Pr}_{\alpha \in L, w \in N_{G'}(\alpha)}[(\alpha, w) \text{ is red}]$. Observe that, by the definition of the set Good, the number of red edges in the graph G' is at most $(1-\gamma)\delta k|\mathit{Good}|$. By Lemma V.E.3 applied to G' with $\eta = (1-\gamma)\delta$, $\beta = \delta/2$, and $\nu = \gamma/2$, we get that this probability is at most $\max\{(1-\gamma)\delta/((1-\gamma/2)(1-\lambda_0/\epsilon)), \delta/2\}$, where $\lambda_0 = \lambda(\delta/2, \gamma/2)$. This is at most $\delta - \gamma\delta/4$ if $\lambda_0/\epsilon < \gamma/4$. □

Finally, we have

$$\mathbf{Pr}_{\alpha \in \{0,1\}^n}[\mathcal{C}(x) \text{ is wrong}] = \frac{1}{2^n}\sum_{\alpha \in H} \mathbf{Pr}[\mathcal{C}(x) \text{ is wrong}] + \frac{1}{2^n}\sum_{\alpha \notin H} \mathbf{Pr}[\mathcal{C}(x) \text{ is wrong}]. \tag{V.4}$$

The first term on the right-hand side of (V.4) is at most $\gamma\delta/5$ by Claim V.F.1. For the second term, we upperbound $\mathbf{Pr}[\mathcal{C}(x) \text{ is wrong}]$ by $\mathbf{Pr}[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp] + \mathbf{Pr}[\mathcal{C}(x) = \perp]$. We know by Claim V.F.2 that, for each $\alpha \notin H$, $\mathbf{Pr}[\mathcal{C}(x) = \perp] \leq \gamma\delta/20$. Thus we get that $\mathbf{Pr}_{\alpha \in \{0,1\}^n}[\mathcal{C}(x) \text{ is wrong}]$ is at most

$$\gamma\delta/5 + \gamma\delta/20 + \frac{1}{2^n}\sum_{\alpha \notin H} \mathbf{Pr}[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp] \leq \gamma\delta/4$$
$$+\mathbf{Pr}_{\alpha \in \{0,1\}^n}[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp],$$

which is at most δ by Claim V.F.3.

V.F.2 Proof without simplifying assumptions

Here we explain how to prove our main theorem without any simplifying assumptions. Since we cannot test membership in the set *Good* of k-tuples where $\bar{C}$ makes fewer than $(1-\gamma)\delta k$ errors, we will make a "soft" (probabilistic) decision of how likely a given k-tuple $\bar{x}$ is in *Good* based on the number of correct answers of $\bar{C}(\bar{x})$ in those $k-1$ positions where we know the α's (since we have generated them ourselves). The fewer errors we see, the more likely we are to believe the answer of $\bar{C}(\bar{x})$ for the position where the real input x was placed.

More precisely, we will use the following subroutine TRS (Trust Reducing Strategy):

> "On inputs $\bar{x} = (x_1, \ldots, x_k)$, $i \in [k]$, and α_j's corresponding to the x_j's for $j \in [k]\setminus\{i\}$, compute the number *err* of errors made by $\bar{C}(\bar{x})$ in positions other than i, that is, $err = |\{j \in [k]\setminus\{i\} \mid \neg R((x_j, \alpha_j), \bar{C}(\bar{x})_j)\}|$. Set $\Delta = err - (1-\gamma)\delta k$. If $\Delta \leq 0$, then output $\bar{C}(\bar{x})_i$ with probability 1. Otherwise, for the parameter $\rho = 1 - \gamma/10$, output $\bar{C}(\bar{x})_i$ with probability ρ^{Δ}, and output $\perp$ with probability $1 - \rho^{\Delta}$."

Now our new randomized algorithm $\mathcal{C}$ is as follows:

> "On input x, choose $k-1$ random tapes $\alpha_1, \ldots, \alpha_{k-1} \in \{0,1\}^n$ uniformly at random. Let $x_1, \ldots, x_{k-1}$ be the puzzles corresponding to the chosen random tapes. Pick $i \in [k]$ at random, and set $\bar{x} = (x_1, \ldots, x_{i-1}, x, x_i, \ldots, x_{k-1})$. Run the subroutine TRS on the inputs $\bar{x}$, i, and $\alpha_1, \ldots, \alpha_{k-1}$. If TRS returns a value $y \neq \perp$, then output y; otherwise repeat with new random α's and i. If no output is produced within $4\ln(20/\gamma\delta)/\epsilon$ iterations, then output $\perp$."

We will analyze this algorithm $\mathcal{C}$ using the λ-sampler G defined at the end of Section V.E.2. Recall that G has as its left vertices all possible n-bit random tapes α, and as its right vertices all possible k-tuples of such tapes. For a left vertex α, its neighbors in G correspond to all possible ways of embedding this α in a k-tuple.

First we will bound the probability of timeout of $\mathcal{C}$. Let x be an input corresponding to the random tape α. The k-tuple $\bar{x} = (x_1, \ldots, x_{i-1}, x, x_i, \ldots, x_{k-1})$

of puzzles constructed by the algorithm $\mathcal{C}$ corresponds to the k-tuple $\bar{\alpha} = (\alpha_1, \dots, \alpha_{i-1}, \alpha, \alpha_i, \dots, \alpha_{k-1})$ of random tapes. If $\bar{\alpha} \in \mathit{Good}$, then the TRS subroutine will return $\bar{C}(\bar{x})_i \neq \perp$ with probability 1. Hence, the probability of timeout on x is at most the probability that $\mathcal{C}$ never samples a neighbor $\bar{\alpha} \in \mathit{Good}$ of α in the graph G. As in the previous subsection, we consider the set $H \subseteq \{0,1\}^n$ of all those left vertices α of G such that α has less than $\epsilon/4$ fraction of its neighbors fall into the set *Good*. We get the following analogs of Claims V.F.1 and V.F.2, with exactly the same proofs.

Claim V.F.4. *The set H has density at most $\gamma\delta/5$.*

Claim V.F.5. *For every $\alpha \notin H$ and the puzzle x corresponding to that random tape α, we have $\mathbf{Pr}[\mathcal{C}(x) = \perp] \leq \gamma\delta/20$, where the probability is over the internal randomness of $\mathcal{C}$.*

Next we analyze the probability of $\mathcal{C}$ outputting a wrong answer, conditioned on its output being something other than $\perp$. For each edge $((\alpha_1, \dots, \alpha_k), \alpha_i)$ of the graph G, we color this edge green if $\bar{C}(x_1, \dots, x_k)_i$ is correct, and we color it red otherwise.

Consider the following random experiment $\mathcal{E}$:

> "Pick a random $\alpha \in L$, and its random incident edge $e = (\alpha, \bar{\alpha})$ in G, for $\bar{\alpha}$ containing α in position $i \in [k]$. Let err be the number of errors made by $\bar{C}(\bar{x})$ in positions other than i, and let $\Delta = \mathit{err} - (1-\gamma)\delta k$. If $\Delta \leq 0$, output the edge e. Otherwise, output e with probability ρ^{Δ} (for $\rho = (1-\gamma/10)$), and output $\perp$ with probability $1 - \rho^{\Delta}$."

For each $\alpha \in \{0,1\}^n$ and the puzzle x corresponding to the random tape α, we have

$$\mathbf{Pr}[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp] = \mathbf{Pr}[\mathcal{E} \text{ outputs red edge incident to } \alpha \mid \mathcal{E} \text{ outputs some edge incident to } \alpha], \tag{V.5}$$

where the first probability is over internal randomness of $\mathcal{C}$, and the second probability is over the random choices of $\mathcal{E}$ for the fixed α (i.e., over the random choice of an edge e incident to α, and the random choice whether e is output).

Rather than analyzing the experiment $\mathcal{E}$, however, we will consider another experiment that is the same as $\mathcal{E}$ except that *err* is defined as the total number of errors made by $\bar{C}(\bar{x})$ in *all* positions (i.e., including the position i). That is, we consider the following experiment $\mathcal{E}'$:

> "Pick a random $\alpha \in L$, and its random incident edge $e = (\alpha, \bar{\alpha})$ in G, for $\bar{\alpha}$ containing α in position $i \in [k]$. Let *err* be the number of errors made by $\bar{C}(\bar{x})$ in all positions, and let $\Delta = err - (1-\gamma)\delta k$. If $\Delta \leq 0$, output the edge e. Otherwise, output e with probability ρ^{Δ} (for $\rho = (1-\gamma/10)$), and output $\perp$ with probability $1-\rho^{\Delta}$."

Claim V.F.6. *For each edge e of G, we have*

$$\mathbf{Pr}[\mathcal{E}' \textit{ outputs } e] \leq \mathbf{Pr}[\mathcal{E} \textit{ outputs } e] \leq (1/\rho)\mathbf{Pr}[\mathcal{E}' \textit{ outputs } e].$$

Proof. Let $p = \mathbf{Pr}[\mathcal{E} \text{ outputs } e]$, and let $p' = \mathbf{Pr}[\mathcal{E}' \text{ outputs } e]$. If edge e is green or $\Delta \leq 0$, then $p' = p$. If e is red and $\Delta > 0$, then $p' = \rho p$. In either case, we have $p' \leq p$ and $p \leq (1/\rho)p'$, as required. □

As a corollary of Claim V.F.6, we get the following.

Claim V.F.7. *For each $\alpha \in \{0,1\}^n$ and the puzzle x corresponding to α, we have that*

$$\mathbf{Pr}[\mathcal{E} \textit{ outputs red edge incident to } \alpha \mid \mathcal{E} \textit{ outputs some edge incident to } \alpha] \leq$$
$$(1/\rho) \cdot \mathbf{Pr}[\mathcal{E}' \textit{ outputs red edge incident to } \alpha \mid \mathcal{E}' \textit{ outputs some edge incident to } \alpha].$$

Proof. The proof is by Claim V.F.6 and the definition of conditional probability. □

Now we can prove the following analog of Claim V.F.3.

Claim V.F.8. $\mathbf{Pr}_{\alpha \in \{0,1\}^n}[\mathcal{C}(x) \textit{ is wrong} \mid \mathcal{C}(x) \neq \perp] \leq \delta - \gamma\delta/4.$

Proof. By (V.5) and Corollary V.F.7, we get that $\mathbf{Pr}_{\alpha\in\{0,1\}^n}[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \bot]$ is at most

$$(1/\rho)\cdot \quad \mathbf{Exp}_{\alpha\in L}[\mathbf{Pr}[\mathcal{E}' \text{ outputs red edge incident to } \alpha \mid \mathcal{E}' \text{ outputs some edge incident to } \alpha]]. \tag{V.6}$$

To upperbound the conditional probability of getting a red edge in the experiment $\mathcal{E}'$, we assign weights to the edges of our graph $G = G(L \cup R, E)$ as follows: An edge $(\alpha, \bar{\alpha}) \in E$ between $\alpha \in L$ and $\bar{\alpha} \in R$ gets the weight $wt(e) = \rho^{\Delta}$, where Δ is as in the definition of the experiment $\mathcal{E}'$ (i.e., Δ is the total number of errors of $\bar{C}(\bar{x})$ minus $(1-\gamma)\delta k$).

For each $\alpha \in L$, let Red_α denote the set of all red edges incident to α, and let

$$\xi(\alpha) = \frac{\sum_{e\in Red_\alpha} wt(e)}{\sum_{e\in E_\alpha} wt(e)},$$

where E_α denotes the set of all edges incident to α. The expectation in (V.6) is exactly $\mu = \mathbf{Exp}_{\alpha\in L}[\xi(\alpha)]$.

Let Red be the set of all red edges in G, and let $\eta = (1/k|R|)\sum_{e\in Red} wt(e)$. Let us partition the set $(\{0,1\}^n)^k$ of k-tuples into the subsets $Good_i$, for $i \geq 0$, where $Good_0 = Good$, and for each $i \geq 1$, $Good_i$ contains all those k-tuples $\bar{\alpha} \in R$ where $\bar{C}(\bar{x})$ makes exactly $((1-\gamma)\delta k + i)$ errors.

Apply Lemma V.E.4 to G, the partitioning $R = \cup_{i\geq 0} Good_i$ with the corresponding weights $\rho^0, \rho^1, \ldots$, and the measure τ of $Good_0$ being at least ϵ. For a parameter β (to be determined later) and $\lambda_0 = \lambda(\beta, \nu)$, we get that μ is at most the maximum of β and the following expression:

$$\frac{|R|\eta}{(1-\nu)(1-\lambda_0/\epsilon)\sum_{i\geq 0}|Good_i|\rho^i}, \tag{V.7}$$

By the definition of the sets $Good_i$, we get that

$$\eta \leq (1/k|R|)\sum_{i\geq 0}|Good_i|((1-\gamma)\delta k + i)\rho^i.$$

Using this bound on η, we can upperbound the expression in (V.7) by

$$\frac{\sum_{i\geq 0} |Good_i|((1-\gamma)\delta + (i/k))\rho^i}{(1-\nu)(1-\lambda_0/\epsilon)\sum_{i\geq 0}|Good_i|\rho^i}. \tag{V.8}$$

For $t = \gamma\delta k/4$, let us split the sum in the numerator of (V.8) into two sums: for $0 \leq i \leq t$ and for $i > t$. We can bound each of these two sums as follows:

$$\begin{aligned}\sum_{i=0}^{t} |Good_i|((1-\gamma)\delta + (i/k))\rho^i &\leq ((1-\gamma)\delta + t/k)\sum_{i=0}^{t}|Good_i|\rho^i \\ &\leq (1-3\gamma/4)\delta\sum_{i\geq 0}|Good_i|\rho^i,\end{aligned}$$

and

$$\sum_{i>t} |Good_i|((1-\gamma)\delta + (i/k))\rho^i \leq \rho^t|R|.$$

Plugging these bounds into (V.8), and recalling that $|Good_0| \geq \epsilon|R|$, we upperbound (V.8) by

$$\frac{1}{(1-\nu)(1-\lambda_0/\epsilon)}\left((1-3\gamma/4)\delta + \frac{\rho^t|R|}{\sum_{i\geq 0}|Good_i|\rho^i}\right) \leq \frac{(1-3\gamma/4)\delta + \rho^t/\epsilon}{(1-\nu)(1-\lambda_0/\epsilon)}.$$

Finally, by (V.6), we get

$$\mathbf{Pr}_{\alpha\in\{0,1\}^n}[\mathcal{C}(x) \text{ is wrong} \mid \mathcal{C}(x) \neq \perp] \leq \max\left\{\frac{(1-3\gamma/4)\delta + \rho^t/\epsilon}{\rho(1-\nu)(1-\lambda_0/\epsilon)}, \frac{\beta}{\rho}\right\},$$

which is at most $\delta - \gamma\delta/4$, for $\rho = 1-\gamma/10$, $\beta = (27/40)\delta$, $\nu = (3/10)\gamma$, $\lambda_0/\epsilon \leq \gamma/20$ and $\rho^t/\epsilon \leq \gamma\delta/100$. □

Now we finish the proof of our main theorem.

Proof of Theorem V.D.1. The proof follows from Claims V.F.4, V.F.5, and V.F.8 in exactly the same way as the proof of the main theorem under the simplifying assumptions, given at the end of Section V.F.1. □

V.G Open problems

While the results here are fairly general, there are some obvious possible extensions. First, can similar results be proved for other domains, such as public-coin protocols [PV07]. Also, our bounds on the adversary's success probability, although asymptotically exponentially small, are quite weak when applied to concrete problems such as actual CAPTCHA protocols with reasonable numbers of repetitions. Can the bounds be improved quantitatively (getting smaller constant in the exponent for ϵ in Theorem V.D.1)? Finally, we would like to find more applications of our results, for example, to such problems as making strong secret agreement protocols from weak ones [Hol05].

Acknowledgements : This Chapter is, in part, based on the paper "Chernoff-type Direct Product Theorems" co-authored with Russell Impagliazzo and Valentine Kabanets. In Advances in Cryptology - CRYPTO 2007, Twenty-Seventh Annual International Cryptology Conference, pages 500–516, 2007.

VI

Applications of Direct Product Theorems in Cryptography

VI.A Introduction

The security of most of the cryptograhic primitives is based on a class of functions called *one-way functions.* These are functions which are easy to compute but hard to invert on the average. This essentially means that for most inputs x, given $f(x)$, it is hard to compute any element of $f^{-1}(f(x))$. So, the parameter of interest is the fraction of inputs for which a function cannot be inverted by efficient algorithms. Given this, a natural question to ask is whether there is a generic method to amplify this parameter. That is, if there is a one-way function which is weak in the sense that it is hard to invert the function for some non-negligible number of inputs for efficient algorithms, then does that imply that there is a strong one-way function with respect to efficient algorithms (hard to invert on almost all inputs)? Moreover, is there a generic way to construct a strong one-way function from weak ones? direct product theorems play a major role in showing that if a one-way function exists with weak parameters, then one way functions exist with stronger parameters.

As we have seen in the previous Chapter, direct product theorems have more immediate use in many cryptographic problems. In this Chapter we will look at more example where the ideas in this Dissertation are useful in solving some cryptographic problems.

VI.B Security Amplification for Cryptographic Primitives

We discuss how techniques from the previous sections can be used to amplify security properties of Message Authentication Codes (MAC) and Digital Signatures (DS). We also discuss security amplification of Pseudorandom Generators (PRG), and Pseudorandom Functions (PRF). In the following subsections, we discuss these primitives in detail. Most of the definitions and terminology used in this section are part of the standard cryptographic literature. Much of the terminology and definition used in this section have been borrowed from [GB01] and [Gol01].

Before we start looking at the specific constructions and proof arguments let us try to understand why such constructions are interesting and useful. Apart from the fact that the details of each of the constructions is different, these are interesting because the proof of security involves showing that an adversary has small chance of success after multiple rounds of interaction with a verifier. In the previous Chapter, we have seen cases where a direct product construction failed to amplify the security of a protocol that involved more than 3 rounds of interaction. So, direct product constructions cannot be used to show security amplification in general. Instead, we have to show this individually for the specific type of protocol.

VI.B.1 Message Authentication Codes (MACs)

The goal of Message Authentication Scheme is to allow a sender to send a message to a receiver such that if the message is corrupted in the route, then the receiver will almost certainly detect this. A Message Authentication Code (MAC) is a special case of Message Authentication Scheme where the sender, given a message, generates a "tag" for the message which is sent along with the message and which serves as the authentication check. More specifically, here is the definition of MAC that we will be working with.

Definition VI.B.1 (MAC, [GB01])**.** A message-authentication code Π consists of three algorithms, $\Pi = (\mathcal{K}; MAC; VF)$, as follows:

- The randomized key generation algorithm $\mathcal{K}$ generates a string K. We let $Keys(\Pi)$ denote the set of all strings that have non-zero probability of being output by $\mathcal{K}$. The members of this set are called keys. We write $K \overset{\$}{\leftarrow} \mathcal{K}$ for the operation of executing $\mathcal{K}$ and letting K denote the key returned.
- The MAC-generation algorithm MAC, which might be randomized [1], takes a key $K \in Keys(\Pi)$ and a plaintext $M \in \{0,1\}^*$ to return a tag $Tag \in \{0,1\}^* \cup \{\perp\}$. We write $Tag \overset{\$}{\leftarrow} MAC_K(M)$ to denote the operation of executing MAC on K and M and letting Tag denote the tag returned.
- The deterministic MAC-verification algorithm VF takes a key $K \in Keys(\Pi)$, a message $M \in \{0,1\}^*$ and a candidate tag $Tag \in \{0,1\}^*$ to return either 1 (accept) or 0 (reject). We write $d \leftarrow VF_K(M, Tag)$ to denote the operation of executing VF on K, M and Tag and letting d denote the decision bit returned.

We require that for any key $K \in Keys(\Pi)$ and any message $M \in \{0,1\}^*$

$$\mathbf{Pr}[Tag = \perp \; OR \; VF_K(M, Tag) = 1 \; : \; Tag \overset{\$}{\leftarrow} MAC_K(M)] = 1.$$

A number $\tau \geq 1$ is called the $tag - length$ associated to the scheme if for any key $K \in Keys(\Pi)$ and any message $M \in \{0,1\}^*$

$$\mathbf{Pr}[Tag = \perp \; OR \; |Tag| = \tau \; : \; Tag \overset{\$}{\leftarrow} MAC_K(M)] = 1.$$

A number $0 < \nu < 1$ is called the *failure probability* associated to the scheme if for any key $K \in Keys(\Pi)$ and any message $M \in \{0,1\}^*$

$$\mathbf{Pr}[Tag = \perp] < \nu.$$

[1] we cannot consider stateful MACs for reasons which will be clear while we discuss the proof

We use the following notion of security for MACs: an adversary A attacking a MAC is said to succeed if it is able to produce a correct message-tag pair after obtaining a number of correct tags for messages of its choice. The probability of success is computed over the random choice of keys. The adversary is allowed bounded number of "signing queries" which means that it is allowed a bounded number of correct tags for messages of its choice (the choice of messages can be adaptive). Furthermore, it is allowed a bounded number of "verification queries" which means that the adversary is allowed a bounded number of attempts to produce a correct message-tag pair with the restriction that this message was not a part of the signing query. Here is a more formal definition of security for message authentication codes.

Definition VI.B.2 (MAC Security, [GB01])**.** Let $\Pi = (\mathcal{K}, MAC, VF)$ be a message authentication code, and let A be an adversary. We consider the following experiment:
Experiment $\mathbf{Exp}_{\Pi}^{uf-cma}(A)$
$K \overset{\$}{\leftarrow} \mathcal{K}$
Run $A^{MAC_K(.),VF_K(.,.)}$
If A made a verification query (M, Tag) such that the following are true
- The verification oracle returned 1
- A did not, prior to making verification query (M, Tag), make signing query M
Then return 1 else return 0

The *uf-cma advantage* of A is defined as

$$Adv_{\Pi}^{uf-cma}(A) = \mathbf{Pr}[\mathbf{Exp}_{\Pi}^{uf-cma}(A) = 1].$$

Given a Message Authentication Code $\Pi = (\mathcal{K}, MAC, VF)$, we consider the following repeated MAC $\Pi_n = (\mathcal{K}_n, MAC_n, VF_n)$.

```
Algorithm $\mathcal{K}^n$
    $K_1 \xleftarrow{\$} \mathcal{K}$
    $\vdots$
    $K_n \xleftarrow{\$} \mathcal{K}$
    $K \leftarrow (K_1, \ldots, K_n)$
    return $K$
```

```
Algorithm $MAC^n_K(M)$
    Parse $K$ as $(K_1, \ldots, K_n)$
    $T_1 \xleftarrow{\$} MAC_{K_1}(M)$
    $\vdots$
    $T_n \xleftarrow{\$} MAC_{K_n}(M)$
    $Tag \leftarrow (T_1, \ldots, T_n)$
    return $Tag$
```

```
Algorithm $VF^n_K(M, Tag)$
    Let $\Theta = (1-\gamma)\delta n$
    Parse $K$ as $(K_1, \ldots, K_n)$
    Parse $Tag$ as $(T_1, \ldots, T_n)$
    $b_1 \leftarrow VF_{K_1}(M, T_1)$
    $\vdots$
    $b_n \leftarrow VF_{K_n}(M, T_n)$
    $m \leftarrow |\{i : b_i = 1\}|$
    If $(m \geq n - \Theta)$ then return 1
      else return 0
```

The following Theorem gives the security amplification for the repeated MAC.

Theorem VI.B.3. *Let A be any adversary attacking Π_n such that*

$$\mathbf{Adv}^{uf-cma}_{\Pi_n}(A) \geq \epsilon, \textit{where } \epsilon = 16 \cdot q_s \cdot e^{-\gamma^2 \delta n/40},$$

while making q_s MAC generation queries, q_v MAC-verification query, and having running time t. Then there exists an adversary B attacking Π such that

$$\mathbf{Adv}^{uf-cma}_{\Pi}(B) \geq (1-\delta),$$

while making $O(q_s^2/\epsilon)$ MAC-generation queries, q_v MAC-verification queries, and having a running time of $O\left((t + n\omega q_s) \cdot (q_s/\epsilon) \cdot \log(1/\gamma\delta)\right)$.[2]

[2] *ω denotes the maximum time to generate a tag given a message and a key for the MAC Π.*

Consider the following adversary B which attacks Π using the adversary A:

Adversary $B^{MAC_K(.)}$

```
00.  Let $\rho = (1 - \gamma/10)$ and $\Theta = (1 - \gamma)\delta n$
01.  $h \leftarrow$ **Pick-h**
02.  Let $P_h$ denote the subset of all messages $M$ such that $h(M) = 0$
03.  Repeat for at most $timeout = O((q_s/\epsilon) \cdot \log(1/\gamma\delta))$ steps:
04.      Pick $K_1, \ldots, K_{n-1} \overset{\$}{\leftarrow} Keys$
05.      Pick $i \overset{\$}{\leftarrow} [1..n]$
06.      $S_v \leftarrow \phi$
07.      Run $A$
08.          When $A$ asks its signing oracle some query $M$
09.              If $M \in P_h$ then return $\perp$
10.              $B$ makes a signing query $M$ to get the tag $T$
11.              $T \leftarrow (MAC_{K_1}(M), .., MAC_{K_{i-1}}(M), T, .., MAC_{K_{n-1}}(M))$
12.              return $T$ to $A$
13.          When $A$ asks its verification oracle some query $(M, Tag)$
14.              $S_v \leftarrow S_v \cup (M, Tag)$
15.              return 0 to A
16.          After $A$ has made all its queries
17.            For each $(M, Tag) \in S_v$
18.              **If** $(M \notin P_h)$ then skip this $(M, Tag)$ pair
19.              Parse $Tag$ as $(T_1, \ldots, T_n)$
20.              $m \leftarrow |\{j : VF_{K_j}(M, T_j) = 1, j \neq i\}|$
21.              **If** $(m \geq n - \Theta)$ then
22.                with prob. 1, $B$ makes a verification query $(M, T_i)$
23.              **else**
24.                with prob. $\rho^{m-\Theta}$, $B$ makes a verification query $(M, T_i)$
25.          If a verification query has already been made then abort
```

```
26.     return $\perp$
```

Pick-h

```
00. Let $\mathcal{H}$ be a pairwise independent family of hash functions
    which maps the message space into $\{0, 1, ..., (2q_s - 1)\}$.
01. Repeat for at most $64q_s^2/\epsilon^2$ times:
02.     $h \xleftarrow{\$} \mathcal{H}$
03.     Let $P_h$ denote the subset of all messages $M$ such that $h(M) = 0$
04.     $count \leftarrow 0$
05.     Repeat for at most $64q_s^2/\epsilon^2$ times:
06.         Pick $K_1, ..., K_n \xleftarrow{\$} Keys$
07.         Run $A$
08.             When $A$ asks a signing query $M$
09.                 If $(M \in P_h)$, then abort and continue with step 5
10.                 else return $Tag \leftarrow (MAC_{K_1}(M), ..., MAC_{K_n}(M))$ to $A$
11.             When $A$ asks a verification query $(M, Tag)$
12.                 If $VF_{(K_1,...,K_n)}(Tag) = 1$ and $M \in P_h$
13.                     increase $count$ by 1 and continue at step 5
14.     If $(count \geq 4q_s/\epsilon)$ then return $h$
```

Intuition behind the proof The proof of the Theorem follows by realizing that the above adversary attacking Π (while using adversary attacking Π_n as an Oracle) is defined on the same lines as the solver for the weaker puzzle (which uses the solver for the harder puzzle as an Oracle) in the proof of Theorem V.D.1 of the previous Chapter. That is, it uses $k-1$ self-generated MAC's to measure the "quality" of attack by A and then use this to make a probabilistic decision for attacking the given MAC. There is just one change in the proof argument. In this case, the adversary B can possibly make "multiple attempts", that is, if it is not able to produce any verification queries with one fixed choice of $K_1, \ldots, K_{n-1}$, then it tries another choice. Let

$$\begin{aligned}
&M_1^1, \ldots, M_{q_s}^1, (M_1^{\prime 1}, Tag_1^1), \ldots, (M_{q_s}^{\prime 1}, Tag_{q_s}^1), \\
&M_1^2, \ldots, M_{q_s}^2, (M_1^{\prime 2}, Tag_1^2), \ldots, (M_{q_s}^{\prime 2}, Tag_{q_s}^2), \\
&\vdots
\end{aligned}$$

be the queries by the adversary A, where each line above denotes the queries corresponding to a choice of $K_1, \ldots, K_{n-1}$. The adversary B has to make sure that a successful verification query should not be one of the signing queries in one of the previous lines. This is handled by randomly partitioning the message space $\mathcal{M}$ into the "attack" messages P, and "query" messages. Here P is a random variable such that any message has probability $1/2q_s$ of falling inside P.

The adversary B aborts its attack if a message that A wishes to sign falls outside of P or if one of $A's$ verification queries falls in P. Due to this, we lose on the success probability of B since there are additional cases where B could abort and hence fail. This is where we lose by a factor of $4q_s$ compared to the Theorem V.D.1. The formal proof of the theorem follows.

Proof. First note that for a randomly chosen $h \overset{\$}{\leftarrow} \mathcal{H}$ (recall $\mathcal{H}$ is a pairwise independent family of hash functions mapping the message space into $[0, ..., (2q_s - 1)]$), P_h is random variable denoting the partition of the message space into two parts which satisfies the following properties:

$$\forall M_1, M_2, \quad \mathbf{Pr}[M_1 \in P_h \mid M_2 \in P_h] = \mathbf{Pr}[M_1 \in P_h] = 1/2q_s \tag{VI.1}$$

For any fixed choice of $\bar{K} = (K_1, ..., K_n)$, let $(M_1^{\bar{K}}, Tag_1^{\bar{K}}), ..., (M_{q_s}^{\bar{K}}, Tag_{q_s}^{\bar{K}})$ denote the signing queries of A. Also, let $(M_V^{\bar{K}}, Tag_V^{\bar{K}})$ denote the first successful verification query, in case A's attack succeeds, and let it denote the last verification query in the case A's attack fails. Furthermore, let $E_{\bar{K}}$ denote the event that $M_1^{\bar{K}}, ..., M_{q_s}^{\bar{K}} \notin P_h$ and $M_V^{\bar{K}} \in P_h$. Next, we bound the probability of the event $E_{\bar{K}}$.

Lemma VI.B.4. $\forall \bar{K} \in G, \quad \mathbf{Pr}_P[E_{\bar{K}}] \geq 1/(4q_s)$.

Proof. We have

$$\begin{aligned}
\mathbf{Pr}[E_{\bar{K}}] &= \mathbf{Pr}[M_V^{\bar{K}} \in P_h \ \& \ \forall i \in [q_s], M_i^{\bar{K}} \notin P_h] \\
&= \mathbf{Pr}[M_V^{\bar{K}} \in P_h] \cdot \mathbf{Pr}[\forall i \in [q_s], M_i^{\bar{K}} \notin P_h \ \mid \ M_V^{\bar{K}} \in P_h] \\
&\geq (1/2q_s) \cdot (1 - \mathbf{Pr}[\exists i \in [q_s], M_i^{\bar{K}} \in P_h \ \mid \ M_V^{\bar{K}} \in P_h]) \\
&\geq (1/2q_s) \cdot \left(1 - \sum_i \mathbf{Pr}[M_i^{\bar{K}} \in P_h \ \mid \ M_V^{\bar{K}} \in P_h]\right) \\
&= (1/2q_s) \cdot \left(1 - \sum_i^{q_s} \mathbf{Pr}[M_i^{\bar{K}} \in P_h]\right) \quad \text{(from (VI.1))} \\
&= 1/4q_s \quad \text{(from (VI.1))}
\end{aligned}$$

□

Let G denote the "good" set corresponding to A's attack, that is, $G = \{(K_1, \ldots, K_n) : \ A's \ attack \ succeeds\}$. We have $Pr[(K_1, \ldots, K_n) \in G] \geq \epsilon$. Given $\bar{K} \in G$, in our construction of adversary B, we abort the attack when event $E_{\bar{K}}$ does not occur. Consider the following random variable:

$$G_{P_h} = \{(K_1, ..., K_n) | (K_1, ..., K_n) \in G \ and \ E_{(K_1, ..., K_n)}\}$$

For our analysis, G_P denotes the "good" set. Furthermore, we have the following

$$\mathbf{Exp}_{P_h}[\mathbf{Pr}_{(K_1,...,K_n)}[(K_1, ..., K_n)] \in G_{P_h}] \geq \epsilon/(4q_s)$$

By averaging, we get that with probability at least $\epsilon/(8q_s)$ over the randomness of P_h there is at least $\epsilon/(8q_s)$ chance that a randomly chosen $(K_1, ..., K_n) \in G_{P_h}$. Let us call such P_h's "good". The subroutine Pick-h with high probability returns an h such that P_h is good. For a fixed good P_h, let *Good* denote the set G_{P_h}. The rest of the analysis is on the lines of the proof of the Theorem V.D.1. So, we define a bipartite graph and then study the special properties of this graph to prove the theorem.

Consider the following bipartite graph $G = G(L \cup R, E)$: the set of left vertices L is the set of keys $K \in Keys(\Pi)$; the right vertices R are all n-tuples of keys $\bar{K} = (K_1, ..., K_n)$; for every $y = (u_1, \ldots, u_k) \in R$, there are n edges $(y, u_1), \ldots, (y, u_n) \in E$. Next, we show that this graph is a *sampler*, which we defined formally in the previous Chapter.

Lemma VI.B.5. *The graph G defined above is a λ-sampler for $\lambda(\mu, \nu) = e^{-\nu^2 \mu k/2}$.*

Proof. This is immediate from Lemma V.E.1. $\square$

Given $\bar{K} = (K_1, ..., K_n)$ such that $E_{\bar{K}}$, let $(M_1, Tag_1), ..., (M_{q_v}, Tag_{q_v})$ be the verification queries of A. Let $(A_{\bar{K}}^{message}, A_{\bar{K}}^{tags})$ be the first correct message-tag pair in the above sequence provided that A makes at least one correct verification query. Otherwise, $(A_{\bar{K}}^{message}, A_{\bar{K}}^{tags})$ is set to be the last (m_i, Tag_i) pair that A produces. In the case that event $E_{\bar{K}}$ does not occur for a given $\bar{K}$, we set $(A_{\bar{K}}^{message}, A_{\bar{K}}^{tags})$ as $(\perp, \perp)$. Note that an adversary can make adaptive verification queries and since we return 0 to all the verification queries of A, we analyze B returning $A_{\bar{K}}^{message}$ and a random element of $A_{\bar{K}}^{tags}$ as its verification query. For further analysis, we denote this output of B as out_B.

First we will bound the probability of timeout of B, that is, out_B is always $\perp$. Let K be the secret key. Let $\bar{K} = (K_1, \ldots, K_{i-1}, K, K_i, \ldots, K_{n-1})$ be the k-tuple of keys constructed by the adversary B. If $\bar{K} \in Good$, then loop 6–25 will return a message-tag pair with probability 1. Hence, the probability of timeout is at most the probability that B never samples a neighbor $\bar{K} \in Good$ of K in the graph G. As in the previous subsection, we consider the set H of all those left vertices K of G such that K has less than $\epsilon'/4$ fraction of its neighbors fall into the set *Good*. We get the following analogs of Claims V.F.1 and V.F.2, with analogous proofs.

Claim VI.B.6. *The set H has density at most $\gamma\delta/5$.*

Proof. Suppose that the density of H is greater than $\beta = \gamma\delta/5$. Let $H' \subseteq H$ be any subset of H of density exactly β. By our assumption, we have that $\mathbf{Pr}_{K \in L, w \in N(K)}[K \in H' \;\&\; w \in Good] < \beta\epsilon'/4$. On the other hand, by Lemma V.E.2

we get that the same probability is at least $\beta(\epsilon' - \lambda_0)/3$ for $\lambda_0 = \lambda(\beta, 2/3)$. This is a contradiction if $\lambda_0 \leq \epsilon'/4$. $\square$

Claim VI.B.7. *For every $K \notin H$, we have $\mathbf{Pr}[out_B = \perp] \leq \gamma\delta/20$, where the probability is over the internal randomness of B.*

Proof. By the definition of H, we get that the probability of timeout on any given $K \notin H$ is at most $(1 - \epsilon'/4)^{4\ln(20/\gamma\delta)/\epsilon'} \leq \gamma\delta/20$. $\square$

Next we analyze the probability of $VF_K(out_B) = 0$, conditioned on the fact that $out_B \neq \perp$.

For each edge $((K_1, \ldots, K_n), K_i)$ of the graph G, we color this edge green if the i^{th} tag in $A^{tags}_{\bar{K}}$ is the correct tag for the message $A^{message}_{\bar{K}}$, and we color it red otherwise.

Consider the following random experiment $\mathcal{E}$:

> "Pick a random $K \in L$, and its random incident edge $e = (K, \bar{K})$ in G, for $\bar{K}$ containing K in position $i \in [n]$. Let err be the number of incorrect tags in $A^{tags}_{\bar{K}}$ for the message $A^{message}_{\bar{K}}$ in positions other than i, and let $\Delta = err - (1-\gamma)\delta n$. If $\Delta \leq 0$, output the edge e. Otherwise, output e with probability ρ^{Δ} (for $\rho = (1 - \gamma/10)$), and output $\perp$ with probability $1 - \rho^{\Delta}$."

For each K, we have

$$\mathbf{Pr}[VF_K(out_B) = 0 \mid out_B \neq \perp] =$$
$$\mathbf{Pr}[\mathcal{E} \text{ outputs red edge incident to } K \mid \mathcal{E} \text{ outputs some edge incident to } K], \tag{VI.2}$$

where the first probability is over internal randomness of B, and the second probability is over the random choices of $\mathcal{E}$ for the fixed K (i.e., over the random choice of an edge e incident to K, and the random choice whether e is output).

Rather than analyzing the experiment $\mathcal{E}$, however, we will consider another experiment that is the same as $\mathcal{E}$ except that err is defined as the total number of incorrect tags of $A^{message}_{\bar{K}}$ in $A^{tags}_{\bar{K}}$ in *all* positions (i.e., including the position i). That is, we consider the following experiment $\mathcal{E}'$:

"Pick a random $K \in L$, and its random incident edge $e = (K, \bar{K})$ in G, for $\bar{K}$ containing K in position $i \in [n]$. Let err be the number of incorrect tags in $A_{\bar{K}}^{tags}$ in all positions, and let $\Delta = err - (1-\gamma)\delta n$. If $\Delta \le 0$, output the edge e. Otherwise, output e with probability ρ^{Δ} (for $\rho = (1-\gamma/10)$), and output $\perp$ with probability $1 - \rho^{\Delta}$."

Claim VI.B.8. *For each edge e of G, we have*

$$\mathbf{Pr}[\mathcal{E}' \textit{ outputs } e] \le \mathbf{Pr}[\mathcal{E} \textit{ outputs } e] \le (1/\rho)\mathbf{Pr}[\mathcal{E}' \textit{ outputs } e].$$

Proof. Let $p = \mathbf{Pr}[\mathcal{E} \text{ outputs } e]$, and let $p' = \mathbf{Pr}[\mathcal{E}' \text{ outputs } e]$. If edge e is green, then $p' = p$. If e is red, then $p' = \rho p$. In either case, we have $p' \le p$ and $p \le (1/\rho)p'$, as required. $\square$

As a corollary of Claim VI.B.8, we get the following.

Claim VI.B.9. *For each K, we have that*

$$\mathbf{Pr}[\mathcal{E} \textit{ outputs red edge incident to } K \mid \mathcal{E} \textit{ outputs some edge incident to } K] \le$$
$$(1/\rho)\cdot\mathbf{Pr}[\mathcal{E}' \textit{ outputs red edge incident to } K \mid \mathcal{E}' \textit{ outputs some edge incident to } K].$$

Proof. The proof is by Claim VI.B.8 and the definition of conditional probability. $\square$

Claim VI.B.10. $\mathbf{Pr}_K[VF_K(out_B) = 0 \mid out_B \neq \perp] \le \delta - \gamma\delta/4$.

Proof. By (VI.2) and Corollary VI.B.9, we get that $\mathbf{Pr}_K[VF_K(out_B) = 0 \mid out_B \neq \perp]$ is at most

$$(1/\rho)\cdot \quad \mathbf{Exp}_{K\in L}[\mathbf{Pr}[\mathcal{E}' \text{outputs red edge incident to } K \mid \mathcal{E}' \text{ outputs some edge incident to } K]]. \qquad \text{(VI.3)}$$

To upperbound the conditional probability of getting a red edge in the experiment $\mathcal{E}'$, we assign weights to the edges of our graph $G = G(L \cup R, E)$ as follows: An edge $(K, \bar{K}) \in E$ between $K \in L$ and $\bar{K} \in R$ gets the weight $wt(e) = \rho^{\Delta}$, where Δ is as in the definition of the experiment $\mathcal{E}'$ (i.e., Δ is the total number of incorrect tags of $A_{\bar{K}}^{message}$ in $A_{\bar{K}}^{tags}$ minus $(1-\gamma)\delta n$).

For each $K \in L$, let Red_K denote the set of all red edges incident to K, and let

$$\xi(K) = \frac{\sum_{e \in Red_K} wt(e)}{\sum_{e \in E_K} wt(e)},$$

where E_K denotes the set of all edges incident to K. The expectation in (V.6) is exactly $\mu = \mathbf{Exp}_{K \in L}[\xi(K)]$.

Let Red be the set of all red edges in G, and let $\eta = (1/n|R|) \sum_{e \in Red} wt(e)$. Let us partition the set the n-tuple of keys into the subsets $Good_i$, for $i \geq 0$, where $Good_0 = Good$, and for each $i \geq 1$, $Good_i$ contains all those n-tuples $\bar{K} \in R$ where there are exactly $((1-\gamma)\delta n + i)$ incorrect tags of $A_{\bar{K}}^{message}$ in $A_{\bar{K}}^{tags}$.

Apply Lemma V.E.4 to G, the partitioning $R = \cup_{i \geq 0} Good_i$ with the corresponding weights $\rho^0, \rho^1, \ldots$, and the measure τ of $Good_0$ being at least ϵ'. We get that μ is at most the maximum of β and the following expression:

$$\frac{|R|\eta}{(1-\nu)(1-\lambda_0/\epsilon') \sum_{i \geq 0} |Good_i| \rho^i}, \tag{VI.4}$$

where $\lambda_0 = \lambda(\beta, \nu)$.

By the definition of the sets $Good_i$, we get that

$$\eta \leq (1/n|R|) \sum_{i \geq 0} |Good_i|((1-\gamma)\delta n + i)\rho^i.$$

Using this bound on η, we can upperbound the expression in (VI.4) by

$$\frac{\sum_{i \geq 0} |Good_i|((1-\gamma)\delta + (i/n))\rho^i}{(1-\nu)(1-\lambda_0/\epsilon') \sum_{i \geq 0} |Good_i| \rho^i}. \tag{VI.5}$$

For $t = \gamma\delta n/4$, let us split the sum in the numerator of (VI.5) into two sums: for $0 \leq i \leq t$ and for $i > t$. We can bound each of these two sums as follows:

$$\begin{aligned} \sum_{i=0}^{t} |Good_i|((1-\gamma)\delta + (i/n))\rho^i &\leq ((1-\gamma)\delta + t/n) \sum_{i=0}^{t} |Good_i| \rho^i \\ &\leq (1 - 3\gamma/4)\delta \sum_{i \geq 0} |Good_i| \rho^i, \end{aligned}$$

and

$$\sum_{i > t} |Good_i|((1-\gamma)\delta + (i/n))\rho^i \leq \rho^t |R|.$$

Plugging these bounds into (VI.5), and recalling that $|Good_0| \geq \epsilon'|R|$, we upper-bound (VI.5) by

$$\frac{1}{(1-\nu)(1-\lambda_0/\epsilon')}\left((1-3\gamma/4)\delta + \frac{\rho^t|R|}{\sum_{i\geq 0}|Good_i|\rho^i}\right) \leq \frac{(1-3\gamma/4)\delta + \rho^t/\epsilon'}{(1-\nu)(1-\lambda_0/\epsilon')}.$$

Finally, by (V.6), we get

$$\mathbf{Pr}_K[VF_K(out_B)=0 \mid out_B \neq \bot] \leq \max\left\{\frac{(1-3\gamma/4)\delta + \rho^t/\epsilon'}{\rho(1-\nu)(1-\lambda_0/\epsilon')}, \frac{\beta}{\rho}\right\},$$

which is at most $\delta - \gamma\delta/4$, for $\rho = 1-\gamma/10$, $\beta = (27/40)\delta$, $\nu = (3/10)\gamma$, $\lambda_0/\epsilon' \leq \gamma/20$ and $\rho^t/\epsilon' \leq \gamma\delta/100$. □

Now we finish the proof of our main Theorem VI.B.3. We have

$$\begin{aligned}\mathbf{Pr}_K[VF_K(out_B)=0] &= \frac{1}{|Keys(\Pi)|}\sum_{K\in H}\mathbf{Pr}[VF_K(out_B)=0] + \\ &\quad \frac{1}{|Keys(\Pi)|}\sum_{K\notin H}\mathbf{Pr}[VF_K(out_B)=0]. \qquad \text{(VI.6)}\end{aligned}$$

The first term on the right-hand side of (VI.6) is at most $\gamma\delta/5$ by Claim VI.B.6. For the second term, we upperbound $\mathbf{Pr}[VF_K(out_B)=0]$ by $\mathbf{Pr}[VF_K(out_B)=0 \mid out_B \neq \bot] + \mathbf{Pr}[out_B = \bot]$. We know by Claim VI.B.7 that, for each $K \notin H$, $\mathbf{Pr}[out_B = \bot] \leq \gamma\delta/20$. Thus we get that $\mathbf{Pr}_K[VF_K(out_B)=0]$ is at most

$$\begin{aligned}\gamma\delta/5 + \gamma\delta/20 \quad + \quad & \frac{1}{|Keys(\Pi)|}\sum_{K\notin H}\mathbf{Pr}[VF_K(out_B)=0 \mid out_B \neq \bot] \leq \\ & \gamma\delta/4 + \mathbf{Pr}_K[VF_K(out_B)=0 \mid out_B \neq \bot],\end{aligned}$$

which is at most δ by Claim VI.B.10.

□

Discussion Following are some important things one should note about security amplification arguments for MACs presented in this section:

1. Note that in the repeated MAC Π_n, the length of the tag as well as the key grows by a factor of n. Ideally, we would like to obtain a security amplification without increasing either of these parameters too much. We hope to address this issue in the future.

2. Another issue to note is that for the case of the deterministic MAC Π, it does not make much sense to use a threshold construction because the receiver always accepts an authenticated message generated by a valid sender. We use threshold construction because apart from being a stronger result, it can be used to show security amplification for the case when the MAC is randomized and there is a small chance that an authenticated message originating from the valid sender is rejected by the receiver.

3. Finally, we note that our arguments do not apply for stateful MACs. This is because we use the attack on Π_n to construct an attack on Π. Here, we use the assumption that for a random choice of keys $(K_1, ..., K_n)$ for Π_n, there is an adversary which succeeds with probability at least ϵ and then we use a bunch of independently chosen keys for Π_n in sequence until we get a good attack on Π. The assumption implicit in this argument is that the probability of success of the adversary attacking Π_n remains at least ϵ for randomly chosen keys for Π_n in a sequence. This might not be true for a stateful MAC.

VI.B.2 Digital Signatures (DSs)

A Digital Signature Scheme is almost the same as Message Authentication Scheme except for the fact that it is a Public Key protocol in contrast with the Message Authentication Scheme which is a Private Key protocol. More specifically, here is a formal definition of a Digital signature Scheme.

Definition VI.B.11 (DS, [GB01])**.** A digital signature scheme $\mathcal{DS} = (\mathcal{K}, Sign, VF)$ consists of three algorithms as follows:

- The randomized key generation algorithm $\mathcal{K}$ (takes no inputs and) returns a pair (pk, sk) of keys, the public key and matching secret key, respectively. We write $(pk, sk) \overset{\$}{\leftarrow} \mathcal{K}$ for the operation of executing $\mathcal{K}$ and letting (pk, sk) be the pair of keys returned.

- The signing algorithm $Sign$ takes the secret key sk and a message M to return a signature or tag $\sigma \in \{0,1\}^* \cup \{\perp\}$. The algorithm may be randomized. We write $\sigma \overset{\$}{\leftarrow} Sign_{sk}(M)$ or $\sigma \overset{\$}{\leftarrow} Sign(sk, M)$ for the operation of running $Sign$ on inputs sk, M and letting σ be the signature returned.
- The deterministic verification algorithm VF takes a public key pk, a message M, and a candidate signature σ for M to return a bit. We write $d \leftarrow VF_{pk}(M,\sigma)$ or $d \leftarrow VF(pk, M, \sigma)$ to denote the operation of running VF on inputs pk, M, σ and letting d be the bit returned.

We require that $VF_{pk}(M,\sigma) = 1$ for any key-pair (pk, sk) that might be output by $\mathcal{K}$, any message M, and any $\sigma \neq \perp$ that might be output by $Sign_{sk}(M)$. Finally, number $0 < \nu < 1$ is called the *failure probability* associated to the scheme if for any key pair (pk, sk) that might be output by $\mathcal{K}$ and any message, we have $\mathbf{Pr}[\sigma = \perp] < \nu$.

The security of digital signatures is defined in a very similar manner as that of message authentication codes.

Definition VI.B.12 (DS, [GB01])**.** Let $\mathcal{DS} = (\mathcal{K}, Sign, VF)$ be a digital signature scheme, and let A be an algorithm that has access to an oracle and returns a pair of strings. We consider the following experiment:
Experiment $\mathbf{Exp}^{uf-cma}_{\mathcal{DS}}(A)$
$(pk, sk) \overset{\$}{\leftarrow} \mathcal{K}$
$(M,\sigma) \leftarrow A^{Sign_{sk}(\cdot)}(pk)$
If the following are true return 1 else return 0:
- $VF_{pk}(M,\sigma) = 1$
- $M \in Messages(pk)$
- M was not a query of A to its oracle

The *uf-cma-advantage* of A is defined as

$$Adv^{uf-cma}_{\mathcal{DS}}(A) = \mathbf{Pr}[\mathbf{Exp}^{uf-cma}_{\mathcal{DS}}(A) = 1].$$

Given a Digital Signature $\mathcal{DS} = (\mathcal{K}, Sign, VF)$, we consider the following repeated DS $\mathcal{DS}_n = (\mathcal{K}^n, Sign^n, VF^n)$.

```
Algorithm $\mathcal{K}^n$
    $K_1 = (pk_1, sk_1) \overset{\$}{\leftarrow} \mathcal{K}$
    $\vdots$
    $K_n = (pk_n, sk_n) \overset{\$}{\leftarrow} \mathcal{K}$
    $K \leftarrow (K_1, \ldots, K_n)$
    return $K$
```

```
Algorithm $Sign^n_K(M)$
    Parse $K$ as $(K_1, \ldots, K_n)$
    $\sigma_1 \overset{\$}{\leftarrow} Sign_{sk_1}(M)$
    $\vdots$
    $\sigma_n \overset{\$}{\leftarrow} Sign_{sk_n}(M)$
    $\sigma \leftarrow (\sigma_1, \ldots, \sigma_n)$
    return $\sigma$
```

```
Algorithm $VF^n_K(M, Tag)$
    Let $\Theta = (1-\gamma)\delta n$
    Parse $K$ as $(K_1, \ldots, K_n)$
    Parse $\sigma$ as $(\sigma_1, \ldots, \sigma_n)$
    $b_1 \leftarrow VF_{pk_1}(M, \sigma_1)$
    $\vdots$
    $b_n \leftarrow VF_{pk_n}(M, \sigma_n)$
    $m \leftarrow |\{i : b_i = 1\}|$
    If $(m \geq n - \Theta)$ then return $1$
      else return $0$
```

The following Theorem gives the security amplification for the repeated DS. The proof is almost the same as for MACs and we omit explicit details.

Theorem VI.B.13. *Let A be any adversary attacking $\mathcal{DS}_n$ such that*

$$\mathbf{Adv}^{uf-cma}_{\mathcal{DS}_n}(A) \geq \epsilon, \textit{where } \epsilon = 16 \cdot q_s \cdot e^{-\gamma^2 \delta n/40},$$

while making q_s signing queries, q_v verification queries, and having running time t. Then there exists an adversary B attacking $\mathcal{DS}$ such that

$$\mathbf{Adv}^{uf-cma}_{\mathcal{DS}}(B) \geq (1-\delta),$$

while making $O(q_s^2/\epsilon)$ signing queries, q_v verification queries, and having a running *time of* $O\left((t + n\omega q_s) \cdot (q_s/\epsilon) \cdot \log\left(1/\gamma\delta\right)\right)$.[3]

VI.B.3 Pseudorandom Generators (PRGs)

Intuitively, pseudorandom generators are functions which expands a truly random string into a random "looking" string. In this subsection we consider Hardness Amplification for pseudorandom generators (PRGs). Given a *weak* pseudorandom generator, we use it to construct a *strong* pseudorandom generator. We start by defining these terms.

Definition VI.B.14 (pseudorandomness)**.** An ensemble $\{X_n\}_{n \in N}$ is called $\delta(n)$-pseudorandom if for every probabilistic polynomial-time algorithm, D, and all sufficiently large n's

$$|Pr[D(X_n, 1^n) = 1] - Pr[D(U_n, 1^n) = 1]| < \delta(n).$$

We will use the following standard definition of a pseudorandom generator.

Definition VI.B.15 (pseudorandomness generator)**.** A $\delta(n)$-pseudorandom generator is a deterministic polynomial-time algorithm, G, satisfying the following two conditions:

1. **expansion**: there exists a function $l : N \rightarrow N$ so that $l(n) > n$ for all $n \in N$, and $|G(s)| = l(|s|)$ for all $s \in \{0,1\}^*$.

2. **pseudorandomness**: the ensemble $\{G(U_n)\}_{n \in N}$ is $\delta(n)$-pseudorandom.

Definition VI.B.16 (weak and strong PRGs)**.** A $\delta(n)$-pseudorandom generator is called *weak* if $\delta(n) = \mathit{constant}$ for all sufficiently large n's.

A $\delta(n)$-pseudorandom generator is called *strong* if $\delta(n) < 1/p(n)$ for any polynomial $p(.)$ and all sufficiently large n's.

[3] *ω denotes the maximum time to sign a message.*

Definition VI.B.17 (Unpredictability)**.** An ensemble $\{X_n\}_{n\in N}$ is called $\delta(n)$ - unpredictable in polynomial time if for every probabilistic polynomial time algorithm A, all $i \in [n]$, and all sufficiently large n's

$$Pr\left[A(1^{|X_n|}, X_n[1..i]) = X_n[i+1]\right] < \frac{1}{2} + \delta(n)$$

where $X_n[n+1]$ denotes a uniformly chosen bit.

The following Theorem shows that under certain conditions, weak pseudorandom generators imply strong ones.

Theorem VI.B.18 (weak PRG implies strong PRG)**.** *Given that there is a weak pseudorandom generator with expansion factor $l(n) = O(n^2)$. Then there exists a strong pseudorandom generator.*

Proof. We start by looking at the construction of the strong generator. We denote the seed of G_{strong} by the tuple (s, r), where s is a string of length kn and r is a string of length k ($k = O(n)$).

```
$G_{strong}(s, r)$
        Parse $s$ into $k(= O(n))$ strings $(s_1, \ldots, s_k)$ of length $n$ each.
        $p \leftarrow 0...0$
        for $i \leftarrow 1$ to $k$
            if $(r[i] = 1)$ then $p \leftarrow bitwiseXOR(p, G_{weak}(s_i))$
```

Note that the the seed for the generator G_{strong} has increased by a factor of k. This is what imposes the constraint that $l(n) > k(n+1) = O(n^2)$. We show that the generator above is a strong generator using the following sequence of arguments: **(i)** use the "pseudorandom versus unpredictability" reduction to argue that $G_{weak}(U_n)$ is hard to predict, **(ii)** use direct product theorem to argue that the product of ensembles $G_{weak}(U_n)$ is even harder to predict, **(iii)** use Goldreich-Levin theorem to argue that the bit-wise XOR of a random subset of these ensembles remains hard to predict, and finally **(iv)** use the "pseudorandom versus unpredictability" reduction to argue that G_{strong} is a strong pseudorandom generator. We present these four steps in the form of the following Claims.

We will need the following Theorem for some of the claims in this subsection.

Theorem VI.B.19 (pseudorandomness versus unpredictability)**.** *If an ensemble* $\{X_n\}_{n \in N}$ *is* $\delta(n)$*-unpredictable, then it is* $(n \cdot \delta(n))$*-pseudorandom. Also, if an ensemble* $\{X_n\}_{n \in N}$ *is* $\delta(n)$*-pseudorandom, then it is* $\delta(n)$*-unpredictable.*

Claim VI.B.20. $\{G_{weak}(U_n)\}_{n \in N}$ *is* δ*-unpredictable.*

Proof. This follows from Theorem VI.B.19. □

The next claim shows that simultaneously predicting the i^{th} bit of k independent instances of $G_{weak}(U_n)$ becomes very hard.

Claim VI.B.21. *For every probabilistic polynomial-time algorithm* A*, for all* $i \in [n]$*, for any polynomial* $q(.)$*, and all sufficiently large* n*'s we have:*

$$\begin{aligned} Pr[A(1^{|k \cdot l(n)|}, G_{weak}(U_{n,1})[1..i], \ldots, G_{weak}(U_{n,k})[1..i]) = \\ (G_{weak}(U_{n,1})[i+1], \ldots, G_{weak}(U_{n,k})[i+1])] < \frac{1}{q(n)}. \end{aligned}$$

Proof. This follows from the fact that predicting the i^{th} bit of a pseudorandom generator can be modeled as a weakly verifiable puzzle and we can then apply the direct product theorem for weakly verifiable puzzles. Here the secret string α is the uniformly chosen string U_n, the puzzle x is the string $G_{weak}(\alpha)[1...(i-1)]$ and the verification procedure for an answer b is $V(\alpha, b) = 1$ if $G_{weak}(\alpha)[i] = b$ and 0 otherwise. We can then apply the Theorem V.D.1 of the previous chapter with the value of $\gamma = 1$. □

We will need the following version of the Goldreich-Levin Theorem for our next claim.

Theorem VI.B.22 (GL89)**.** *There is a probabilistic algorithm A with the following property. Let $h \in \{0,1\}^k$ be any string , and let $B : \{0,1\}^k \rightarrow \{0,1\}$ be any predicate such that $|Pr_{r\in\{0,1\}^n}[B(r) = \langle h, r\rangle] - 1/2| \geq \gamma$ for some $\gamma > 0$. Then, given oracle access to B and given γ, the algorithm A runs in time poly$(k, 1/\gamma)$, and outputs a list of size $l = O(1/\gamma^2)$ such that with high probability the string is on this list.*

The following claim shows that predicting the bit given by taking the inner product of the i^{th} bit of k independent instances of $G_{weak}(U_n)$ with a random string $r \in \{0,1\}^k$.

Claim VI.B.23. *For every probabilistic polynomial-time algorithm B, all $i \in [n]$, for all polynomial $q(.)$, for all sufficiently large n's, and $r \in_u \{0,1\}^k$ we have*

$$Pr[A(1^{|k\cdot l(n)|}, G_{weak}(U_{n,1})[1..i], \ldots, G_{weak}(U_{n,k})[1..i]) = \\ \langle G_{weak}(U_{n,1})[i+1], \ldots, G_{weak}(U_{n,k})[i+1], r\rangle] < \frac{1}{2} + \frac{1}{q(n)}.$$

Proof. The above theorem follows from the Claim VI.B.30 and Theorem VI.B.22.

□

Finally, we use the "unpredictability versus pseudorandomness" reduction to show that G_{strong} satisfies a stronger psedorandomness property.

Claim VI.B.24. *G_{strong} is a strong pseudorandom generator.*

Proof. This follows from Theorem VI.B.19. □

□

VI.B.4 Pseudorandom Functions (PRFs)

In this subsection, we talk about hardness amplification of Pseudorandom functions. We will use the same set of ideas as in the previous section to show that a weak pseudorandom function implies a strong one. Let us start defining these terms. Consider the following definition of pseudorandom function ensemble borrowed from [Gol01].

Definition VI.B.25 (Efficiently computable pseudorandom function)**.** Let $d, r : \mathbb{N} \to \mathbb{N}$. We say that

$$\{f_s : \{0,1\}^{d(|s|)} \to \{0,1\}^{r(|s|)}\}_{s\in\{0,1\}^*}$$

is an efficiently computable generalized pseudorandom function ensemble if the following two conditions hold

1. (efficient evaluation): There exists a polynomial time algorithm that on inputs s and $x \in \{0,1\}^{d(|s|)}$ returns $f_s(x)$.
2. (pseudorandomness): For every probabilistic polynomial-time oracle machine M, every polynomial $p(.)$ and all sufficiently large n's

   $$|\mathbf{Pr}[M^{F_n}(1^n) = 1] - \mathbf{Pr}[M^{H_n}(1^n) = 1]| < 1/p(n)$$

   where F_n is a random variable uniformly distributed over the multi-set $\{f_s\}_{s\in\{0,1\}^n}$, and H_n is uniformly distributed among all functions mapping $d(n)$-bit long strings to $r(n)$-bit long strings.

Definition VI.B.26 (Weak and strong pseudorandom function)**.** A weak pseudorandom function is a pseudorandom function where the distinguishing probability is $1/p(n) = 1/c$ for some constant c.

A strong pseudorandom function is a pseudorandom function where the distinguishing probability is $< p(n)$ for any polynomial $p(.)$ and all sufficiently large n's.

Definition VI.B.27 (Unpredictability for pseudorandom functions)**.** Let $\delta : \mathbb{N} \to \mathbb{N}$. A pseudorandom function ensemble $\{F_n\}_{n\in\mathbb{N}}$ is called a $\delta(n)$ unpredictable if for all sufficiently large n and any polynomial time algorithm algorithm A we have

$$\forall i, \mathbf{Pr}_{f\in F_n, x\in\{0,1\}^{d(n)}}[A(1^n, f(x)[1...i-1]) = f(x)[i]] < \delta(n).$$

Theorem VI.B.28 (Weak PRF implies strong PRF)**.** *Given that there is a weak pseudorandom function $\{F_n\}_{n\in\mathbb{N}}$, then there is a strong pseudorandom function $\{F'_n\}_{n\in\mathbb{N}}$.*

Proof. Here the description of the construction of the strong pseudorandom function from a weak one:

> F'_n: Given $s_1, ..., s_k, t \in \{0,1\}^k$ and $x \in \{0,1\}^{d(|s_i|)}$ the value of the function $f_{(s_1,...,s_k,t)}(x)$ is defined as $\forall i, f_{(s_1,...,s_k,t)}(x)[i] = \langle (f_{s_1}(x)[i], ..., f_{s_k}(x)[i]), t\rangle$

Note that the descirption length of the function has increased by a factor of k. We show that the function above is a strong psedorandom function using the following sequence of arguments: **(i)** use the "pseudorandom versus unpredictability" reduction to argue that $f_s(U_{d(n)})$ is hard to predict, **(ii)** use direct product theorem to argue that the product of ensembles $f_{(s_1,...,s_k)}(U_{d(n)})$ is even harder to predict, **(iii)** use Goldreich-Levin theorem to argue that the bit-wise XOR of these functions remains hard to predict, and finally **(iv)** use the "pseudorandom versus unpredictability" reduction to argue that F_n is a strong pseudorandom function. We present these four steps in the form of the following Claims.

We will need the following Theorem for some of the claims in this subsection.

Claim VI.B.29. $\{F_n(U_{d(n)})\}_{n\in N}$ *is δ-unpredictable for some constant δ.*

Proof. This follows from Theorem VI.B.19. □

The next claim shows that simultaneously predicting the i^{th} bit of k independent instances of $F_n(U_{d(n)})$ becomes very hard.

Claim VI.B.30. *For every probabilistic polynomial-time algorithm A, for all $i \in [r(n)]$, for any polynomial $q(.)$, and all sufficiently large n's we have:*

$$\mathbf{Pr}[A(1^{k\cdot n}, F_n^1(U_{d(n)}^1)[1..i], \ldots, F_n^k(U_{d(n)}^k)[1..i]) = \\ (F_n^k(U_{d(n)}^k)[i+1], \ldots, F_n^k(U_{d(n)}^k)[i+1])] < \frac{1}{q(n)},$$

where $F_n^1, ..., F_n^k$ denotes independent instances of F_n and $U_{d(n)}^1, ..., U_{d(n)}^k$ denotes independent instances of $U_{d(n)}$.

Proof. This follows from the fact that predicting the i^{th} bit of a pseudorandom function can be modeled as a weakly verifiable puzzle and we can then apply direct product theorem for weakly verifiable puzzles. Here the secret string α is the uniformly chosen function $f \in F_n$ plus uniformly chosen string $w \in U_n$, the puzzle x is the string $f(w)[1...(i-1)]$ and the verification procedure for an answer b is $V(\alpha, b) = 1$ if $f(w)[i] = b$ and 0 otherwise. We can then apply the Theorem V.D.1 of the previous chapter with the value of $\gamma = 1$ to obtain the above Theorem. $\square$

Claim VI.B.31. *For every probabilistic polynomial-time algorithm B, all $i \in [n]$, for all polynomial $q'(.)$, for all sufficiently large n's, we have*

$$Pr[A(1^{k \cdot n}, F_n^1(U_{d(n)}^1)[1..i], \ldots, F_n^k(U_{d(n)}^k)[1..i]) = \\ F_n(U_{d(n)}^1)[i+1] \oplus ... \oplus F_n(U_{d(n)}^k)[i+1]] < \frac{1}{2} + \frac{1}{q'(n)},$$

where $F_n^1, ..., F_n^1$ denotes independent instances of F_n and $U_{d(n)}^1, ..., U_{d(n)}^k$ denotes independent instances of $U_{d(n)}$.

Proof. The above theorem follows from the Claim VI.B.30 and Theorem VI.B.22. $\square$

Finally, we use the "unpredictability versus pseudorandomness" reduction to show that F_n' satisfies a stronger psedorandomness property.

Claim VI.B.32. *F_n' is a strong pseudorandom function.*

Proof. This follows from Theorem VI.B.19. $\square$

$\square$

VII

Conclusions and Open Problems

In this Dissertation, we have studied direct product theorems. We have obtained a uniform version of the classical direct product theorem. We also studied a stronger and more general direct product theorem statement which establishes a concentration bound on the fraction of correctly solved problems instances from a pool of independently chosen instances for a given average-case hard problem. The showed this statement for a very generic setting which allowed us to extend the techniques to show security amplification using parallel repetition in various cryptographic protocols.

A number of interesting problems remain open. In Chapter III we obtained a derandomized direct product theorem in the uniform setting but with parameters which fell short of the ideal case. Can we obtain a derandomized version of the theorem in the uniform setting with better parameters (getting the error term $\epsilon = e^{-\Omega(n)}$ instead of the current $\epsilon = e^{-\Omega(\sqrt{n})}$)? Another interesting question is whether we can obtain a stronger "Chernoff-type" theorem in the uniform setting? In Chapter V, we studied Chernoff-type direct product theorems. The parameter ϵ (currently $e^{-\gamma^2 \delta k/40}$) fell short of the information-theoretically optimal ($e^{-\gamma^2 \delta k/2}$). It would be interesting to see one can obtain the result with optimal parameters. In Chapter VI, we looked at Security amplification for MACs by using repetition. This, however, increases the length of the shared secret key as well as the length of the MAC by a factor of k. Ideally, we would like to get security amplification without too much increase in the size of the shared key or MAC.

Another very interesting and related area that we did not touch upon in this Dissertation is direct product testing. Following is an informal description of the problem. Given oracle access to a circuit $C : \{0,1\}^{nk} \rightarrow \{0,1\}^k$, the goal is to differentiate between the following two cases:

1. There exists a function $f : \{0,1\}^n \rightarrow \{0,1\}$ such that $C = f^k$, that is,

$$\forall (x_1, ..., x_k) \in \{0,1\}^{nk}, C(x_1, ..., x_k) = f^k(x_1, ..., x_k)^1$$

2. For any function $f : \{0,1\}^n \rightarrow \{0,1\}$,

$$\mathbf{Pr}_{(x_1,...,x_k)}[C(x_1, ..., x_k) = f^k(x_1, ..., x_k)] < \epsilon$$

A q-query test for the above problem requires to differentiate between the above two cases by querying the circuit C on at most q inputs. In a recent development [DG08], an optimal two query test has been shown for $\epsilon = \Omega(1/k)$. It would be interesting to see a (> 2)-query test which achieves stronger error parameter ϵ.

[1]this can be relaxed to approximate computing.

Bibliography

[ABHL03] L. Vin Ahn, M. Blum, N.J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. *In Advances of Cryptology - EUROCRYPT 2003*, pages 294–311, 2003.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, 160(2):781–793, 2004.

[AS03] S. Arora and M. Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365426, 2003.

[BDCGL92] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average-case complexity. *Journal of Computer and System Sciences*, 44(2):193–219, 1992.

[BFNW93] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[BGG93] M. Bellare, O. Goldreich, and S. Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3:319–354, 1993.

[BIN97] M. Bellare, R. Impagliazzo, and M. Naor. Does parallel repetition lower the error in computationally sound protocols? *Proceeding of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pages 374–383, 1997.

[BOS06] J. Buresh-Oppenheim and R. Santhanam. Making hard problems harder. *In Proceedings of the 21st Annual IEEE Conference on Computational Complexity*, pages 73–87, 2006.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1):1–106, 2006.

[CHS05] R. Canetti, S. Halevi, and M. Steiner. Hardness amplificatio of weakly verifiable puzzles. *In Theory of Cryptography (TCC 2005)*, pages 17–33, 2005.

[DG08] Irit Dinur and Elazar Goldenberg. Locally testing direct products in the high error range. *FOCS'08 (to appear)*, 2008.

[DR06] I. Dinur and O. Reingold. Assignment testers: Towards a combinatorial proof of the pcp theorems. *SIAM Journal on Computing*, 36(4):975–1024, 2006.

[GB01] Shafi Goldwasser and Mihir Bellare. *Lecture Notes in Cryptography.* 2001.

[GGH$^+$07] S. Goldwasser, D. Gutfreund, A. Healy, T. Kaufman, and G. N. Rothblum. Verifying and decoding in constant depth. *In Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 440–449, 2007.

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[GNW95] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR-Lemma. *Electronic Colloquium on Computational Complexity*, (TR95-050), 1995.

[Gol01] O. Goldeich. *Foundations of Cryptography: Volume 1, Basis Tools.* Cambridge University Press, 2001.

[GS00] O. Goldreich and S. Safra. A combinatorial consistency lemma with application to proving the pcp theorem. *SIAM Journal on Computing*, 29(4):1132–1154, 2000.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Journal*, pages 13–30, 1963.

[Hol05] T. Holenstein. Key agreement from weak bit agreement. *In Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 664–673, 2005.

[HVV04] A. Healy, S. Vadhan, and E. Viola. Using nondeterminism to amplify hardness. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, pages 192–201, 2004.

[IJK06] Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Approximately list decoding direct product codes and uniform hardness amplification. *In Proceeding of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 187–196, 2006.

[IJK07] Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Chernoff-type direct product theorems. *In Advances of Cryptology*

- *CRYPTO 2007, 27th Annual International Cryptology Conference*, pages 500–516, 2007.

[IJKW08] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: simplified, optimized and derandomized. *In Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 579–588, 2008.

[Imp95a] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the Thirty-Sixth Annual IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[Imp95b] Russell Impagliazzo. A personal view of average-case complexity. *Proceeding of the Tenth Annual Structure in Complexity Theory Conference*, pages 134–147, 1995.

[Imp02] Russell Impagliazzo. Hardness as randomness: a survey of universal derandomization. *Proceedings of the ICM, Beijing 2002*, 3:659–672, 2002.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[Kab02] Valentine Kabanets. Derandomization: A brief overview. *Bulletin of the European Association for Theoretical Computer Science*, 76:88–103, 2002.

[Lev86] L. A. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986.

[Lev87] L. A. Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.

[O'D04] R. O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69(1):68–94, 2004. (preliminary version in STOC'02).

[PV07] R. Pass and M. Venkitasubramanium. An efficient parallel repetition theorem for arthur-merlin games. *In Proceeding of the 39th Annual ACM Symposium on Theory of Computing*, pages 420–429, 2007.

[PW07] K. Pietrzak and D. Wikstrom. Parallel repetition of computationally sound protocols revisited. *In Theory of Cryptography (TCC 2007)*, pages 86–102, 2007.

[RS42] J. Riordan and C. E. Shannon. The number of two terminal series parallel networks. *Journal of Math. Physics*, 21:83–93, 1942.

[RS97] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. *In Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 475–484, 1997.

[STV01] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001. (preliminary version in STOC'99).

[Tre03] L. Trevisan. List-decoding using the XOR lemma. In *Proceedings of the Forty-Fourth Annual IEEE Symposium on Foundations of Computer Science*, pages 126–135, 2003.

[Tre05] L. Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, pages 31–38, 2005.

[TV02] L. Trevisan and S. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the Seventeenth Annual IEEE Conference on Computational Complexity*, pages 103–112, 2002.

[Yao82] A. C. Yao. Theory and applications of trapdoor functions. In *Proceedings of the Twenty-Third Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[Zuc97] D. Zuckerman. Randomness-optimal oblivious sampling. *Random Structures and Algorithms*, 11(4):345–367, 1997.