Learning Noisy Characters, Multiplication Codes, and Cryptographic Hardcore Predicates

by

Adi Akavia

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2008

Author . . . .
Department of Electrical Engineering and Computer Science
January 30, 2008

Certified by . . . .
Shafi Goldwasser
RSA Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Terry P. Orlando
Chairman, Department Committee on Graduate Students

Learning Noisy Characters, Multiplication Codes, and Cryptographic Hardcore Predicates

by

Adi Akavia

Submitted to the Department of Electrical Engineering and Computer Science
on January 30, 2008, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

We present results in cryptography, coding theory and sublinear algorithms.

In *cryptography*, we introduce a unifying framework for proving that a Boolean predicate is hardcore for a one-way function and apply it to a broad family of functions and predicates, showing new hardcore predicates for well known one-way function candidates such as RSA and discrete-log as well as reproving old results in an entirely different way. Our proof framework extends the list-decoding method of Goldreich and Levin [38] for showing hardcore predicates, by introducing a new class of error correcting codes and new list-decoding algorithm we develop for these codes.

In *coding theory*, we introduce a novel class of error correcting codes that we name: *Multiplication codes (MPC)*. We develop decoding algorithms for MPC codes, showing they achieve desirable combinatorial and algorithmic properties, including: (1) binary MPC of constant distance and exponential encoding length for which we provide efficient *local list decoding* and *local self correcting* algorithms; (2) binary MPC of constant distance and polynomial encoding length for which we provide efficient *decoding* algorithm in random noise model; (3) binary MPC of *constant rate and distance*. MPC codes are unique in particular in achieving properties as above while having a large group as their underlying algebraic structure.

In *sublinear algorithms*, we present the SFT algorithm for finding the sparse Fourier approximation of complex multi-dimensional signals in time logarithmic in the signal length. We also present additional algorithms for related settings, differing in the model by which the input signal is given, in the considered approximation measure, and in the class of addressed signals. The sublinear algorithms we present are central components in achieving our results in cryptography and coding theory. Reaching beyond theoretical computer science, we suggest employing our algorithms as tools for performance enhancement in data intensive applications, in particular, we suggest replacing the $O(N \log N)$-time FFT algorithm with our $\widetilde{\Theta}(\log N)$-time SFT algorithm for settings where a sparse approximation suffices.

Thesis Supervisor: Shafi Goldwasser
Title: RSA Professor of Electrical Engineering and Computer Science

Acknowledgments

My deepest thanks are to Shafi Goldwasser who has been a wonderful advisor, mentor and friend in all these years since I first set foot in her office. I feel blissed and privileged to have had her guidance through the paths of research. Her insights and perspectives on research along with her determination in conducting one have taught me invaluable lessons. Her kindness, support and encouragement while being my advisor have been indispensable. No words can express how indebted I am to her.

Much of this thesis is due to a joint work with Shafi and with Muli Safra; I'm indebted to Shafi and to Muli for their vital contribution to this work.

Parts of this thesis are due to a joint work with Ramarathnam Venkatesan; it's a pleasure to thank Venkie for these parts in particular and for his collaboration as a whole.

I am indebted to Oded Goldreich and to Vinod Vaikuntanathan for endless discussions influencing my entire academic education and this dissertation in particular.

I benefited a great deal from discussions with Nati Linial, Alon Rosen, Adi Shamir, Madhu Sudan, Salil Vadhan, Avi Wigderson, as well as with the fellow students of the theory lab at MIT – many thanks to you all.

I am grateful to my thesis committee members Ron Rivest and Madhu Sudan.

Finally, many thanks to Sara, Marty and Erik, Michal, Yonatan, Alon and Yuval for being my home away from home; to Eema and Aba, Uri, Tamar, Ella and the rest of the family for their endless support and encouragement; and last but not least, many many thanks to Doron for his constant support and love even during the long months I disappeared into this dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis presents results in three areas: cryptography, coding theory and sublinear algorithms.

In *cryptography*, we introduce a unifying framework for proving that a Boolean predicate is hardcore for a one-way function and apply it to a broad family of functions and predicates, showing new hardcore predicates for well known one-way function candidates such as RSA and discrete-log as well as reproving old results in an entirely different way. Our proof framework extends the list-decoding method of Goldreich and Levin [38] for showing hardcore predicates, by introducing a new class of error correcting codes and new list-decoding algorithm we develop for these codes.

In *coding theory*, we introduce a novel class of error correcting codes that we name: *Multiplication codes (MPC)*. We develop decoding algorithms for MPC codes, showing they achieve desirable combinatorial and algorithmic properties, including: (1) binary MPC of constant distance and exponential encoding length for which we provide efficient *local list decoding* and *local self correcting* algorithms; (2) binary MPC of constant distance and polynomial encoding length for which we provide efficient *decoding* algorithm in random noise model; (3) binary MPC of *constant rate and distance*. MPC are unique in particular in achieving properties as above while having no underlying field structure, but rather only a group structure.

In *sublinear algorithms*, we present the SFT algorithm for finding the sparse Fourier approximation of complex multi-dimensional signals in time logarithmic in the signal length. We also present additional algorithms for related settings, differing in the model by which the input signal is given, in the considered approximation measure, and in the class of addressed signals. The sublinear algorithms we present are central components in achieving our results in cryptography and coding theory. Reaching beyond theoretical computer science, we suggest employing our algorithms as tools for performance enhancement in data intensive applications, in particular, we suggest replacing the $O(N \log N)$-time FFT algorithm with our $\widetilde{\Theta}(\log N)$-time SFT algorithm for settings where a sparse approximation suffices.

Organization of this chapter. In section 1.1 we introduce our work on cryptographic hardcore predicates. In section 1.2 we introduce our study of Multiplication codes. In section 1.3 we define the problem of Learning Characters with Noise (LCN) and introduce our results in its study, and, in particular, the SFT algorithm. In section 1.4 we present applications of the SFT algorithm in data intensive algorithms. Finally, in section 1.5 we conclude and describe the organization of the rest of this thesis.

1.1 Cryptographic Hardcore Predicates

Let f be a one-way function, namely a function which is easy to compute, but hard to invert on all but a negligible fraction of its inputs. We say that a Boolean predicate P is a *hardcore predicate for* f if $P(x)$ is easy to compute given x, but hard to guess with non-negligible advantage beyond 50% given only $f(x)$. The notion of hardcore predicates was introduced and investigated in [18, 41] and has since proven central to cryptography and pseudo-randomness.

The standard proof methodology for showing P is hardcore for f is by a reduction from inverting f to predicting P. That is, demonstrate an efficient inversion algorithm for f, given access to a probabilistic polynomial time magic-algorithm B that on input $f(x)$ guesses $P(x)$ with non-negligible advantage over a random guess. Since f is assumed to be a one-way function, it follows that no such algorithm B exists and P is a hardcore predicate.

Blum and Micali [18] were the first to show a hardcore predicates for a function widely conjectured to be one-way. Let p be a prime and g a generator for Z_p^*. The function $EXP_{p,g}\colon Z_{p-1} \to Z_p^*$, $EXP_{p,g}(x) = g^x \bmod p$ is easy to compute and as hard to invert as solving discrete logarithm modulo a prime p. Blum and Micali [18] define the predicate $BM_{p,g}(x) = 0$ if $0 \leq x < \frac{p-1}{2}$ and 1 otherwise, and prove it is a hardcore predicate for $EXP_{p,g}$ if the discrete logarithm problem is intractable. In subsequent years, it was shown for other conjectured one-way functions f and other predicates P, that P is a hardcore predicates for f [5, 16, 32, 41, 42, 51, 59, 86, 90]. Most notably, for the RSA [75] function $RSA\colon Z_n^* \to Z_n^*$, $RSA(x) = x^e \bmod n$, the predicates $P_i(x) = i^{th}$ bit of x were shown hardcore, first for $i = 1, |n|$ [5] and recently for any $1 \leq i \leq |n|$ [51].

Goldreich and Levin [38] address the general question of whether every one-way function (OWF) has some hardcore predicate. They show that for any OWF $f\colon \{0,1\}^n \to \{0,1\}^*$ one can define another OWF $f'\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^* \times \{0,1\}^n$ by $f'(x,r) = (f(x), r)$, so that the predicate $GL(x,r) = \sum_{i=1}^n x_i r_i$ is a hardcore predicates for f'.

The work of Goldreich-Levin, which explicitly addressed hardcore predicates for arbitrary one-way functions, by way of solution gave a polynomial time list-decoding algorithm for a well known error correcting code – the Hadamard code. It introduced an interesting connection between hardcore predicatess and list decoding which, as pointed out by Impagliazzo and Sudan [56, 82, 87], could potentially lead to a general *list decoding methodology* for proving hardcore predicates for one-way functions.

We formalize such a methodology. Given a function f and predicate P, one would have to:

1. **Define a Code.** Identify an error-correcting code $\mathcal{C}^P$ encoding distinct

x's, such that given only $f(x)$ and the capability to compute $P(z)$ on input z, one can query-access the codeword for x, C_x^P. In the case of [38] the code defined was the Hadamard code which is a natural choice as $GL(x,r)$ is precisely the r-th entry of the Hadamard encoding of string x.

2. List Decode. Show a polynomial-time list-decoding algorithm for the code $\mathcal{C}^P$, which works with query access to a corrupted codeword and tolerates a $< \frac{1}{2} - \varepsilon$ fraction of error. In the case of Hadamard code, [38] indeed provides such a list decoding algorithm.

3. Show that predicting P implies access to a corrupted codeword. Show that if there exists a magic algorithm B that on input $f(x)$ predicts $P(x)$ with non-negligible advantage, then there exists an access algorithm which for a non-negligible fraction of x's, on input $f(x)$, can query access a corrupted codeword of x with $< \frac{1}{2} - \varepsilon$ fraction of errors.

Putting all these together, implies that if B exists then f can be inverted for a non-negligible fraction of x's (using the list decoding algorithm). Thus, under the assumption that f is a one-way function, no such B can exist and predicate P is a hardcore predicate.

This is no doubt an elegant methodology but is it a **useful** methodology for proving hardcore predicate results for natural f's and P's? At the very least, can we define appropriate codes and corresponding list decoding algorithms so as to employ this methodology for proving existing hardcore predicate results of [18] for the EXP function, and the results of of [5, 51] for the RSA function?

These questions are the starting point for our investigation.

Our Work

We introduce a unifying framework for proving that predicate P is hardcore for a one-way function f, and apply it to a broad family of functions and predicates, showing new hardcore predicates for well known one-way function candidates as well as reproving old results in an entirely different way.

Our framework follows a list decoding methodology. Thus, the technical essence of the new proofs is to define appropriate codes and to list decode them. These two tasks are independent of the one-way function in question and depend only on the predicate. The only consideration given to the one-way function is in devising a way to access the corrupted codeword.

Our proof framework extends the list-decoding method of Goldreich and Levin [38] for showing hardcore predicates, by introducing a new class of error correcting codes, which we name: *Multiplication Codes (MPC)*, and a new list-decoding algorithm we develop for these codes. In our framework, each predicate corresponds to one error correcting code in the class of MPC, predicting a predicate corresponds to access to a corrupted codeword, and the task of inverting one-way functions corresponds to the task of list decoding a corrupted codeword. A characteristic of the error correcting codes which emerge and are addressed by our framework is that codewords can be approximated by a small number of significant coefficients in their Fourier representation. Moreover, as long as corrupted words are close enough to legal codewords, they share a significant Fourier coefficient. To list decode, we first devise a learning algorithm applied to corrupted codewords for finding their heavy Fourier coefficients, and then find all codewords for which these coefficients are heavy.

Let us now elaborate on the hardcore predicates and one-way functions for which we apply our framework.

Segment Predicates: A Class of New Hardcore Predicates

We apply our framework to prove that a wide class of predicates called *segment predicates* are hardcore predicates for various well known candidate one-way functions.

A segment predicate is any arbitrary assignment of Boolean values to an arbitrary partition of $\mathbb{Z}_N$ into $poly(\log N)$ segments, or a multiplicative shift of such an assignment. A segment predicate can be balanced (with the same number of 0's and 1's) or unbalanced as long as it is far from a constant function. In the latter case of unbalanced predicates, we naturally adapt the definition of hardcore unbalanced predicates to be that it is impossible to compute the predicate better than guessing it at random from $f(x)$.

We prove that any segment predicate is hardcore for any one-way function f defined over $\mathbb{Z}_N$ for which, for a non-negligible fraction of the x's, given $f(x)$ and y, one can efficiently compute $f(xy)$ (where xy is multiplication in $\mathbb{Z}_N$). This includes the functions $EXP_{p,g}$, $RSA(x)$, $Rabin(x) = x^2 \bmod n$, and $ECL_{a,b,p,Q} = xQ$ where Q is a point of high order on an elliptic curve $E_{a,b,p,Q}(Z_p)$ (naturally the appropriate N in each case differs).

In particular, this implies that for every i the *i-partition-bit* is a hardcore predicate for the RSA function where we define the i-th partition bit of x as 0 if $0 \leq 2^i x \leq \frac{N}{2} \bmod N$ and 1 otherwise.

In contrast with the notion of segment predicates, we remark that in the past most all predicates investigated correspond in a fairly direct way with bits in the inverse of $f(x)$. An exception is the work of [69] showing that $P_i(x) = ith$ bit of $ax + b \bmod p$ for randomly chosen a, b, p are hardcore predicates for one-way functions f.

Definition 1.1 (Segment Predicate)**.** *Let $\mathcal{P} = \{P_N \colon \mathbb{Z}_N \to \{\pm 1\}\}$ be a collection of predicates that are non-negligibly far from constant, namely, $\exists$ non-negligible function ρ s.t. $\mathsf{maj}_{P_N} \leq 1 - \rho(k)$ where $k = \log N$.*

- *We say that P_N is a* basic t-segment predicate *if $P_N(x+1) \neq P_N(x)$ for at most t x's in $\mathbb{Z}_N$.*

- *We say that P_N is a* t-segment predicate *if there exist a basic t-segment predicate P' and $a \in \mathbb{Z}_N$ which is co-prime to N s.t. $\forall x \in \mathbb{Z}_N, P_N(x) = P'(x/a)$.*

- *If $\forall N$, P_N is a $t(N)$-segment predicate, where $t(N)$ is polynomial in* $\log N$*, we say that $\mathcal{P}$ is a collection of* segment predicates*.*

Theorem 1.2. *Let $\mathcal{P} = \{P_N \colon \mathbb{Z}_N \to \{\pm 1\}\}$ be a collection of segment predicates. Then, $\mathcal{P}$ is hardcore for RSA, $Rabin$, EXP, ECL, under the assumption that these are OWFs.*

New Proofs of Old Results

It is easy to see that the hardcore predicates of [5, 18, 59] for candidate one-way functions EXP, RSA, $Rabin$ and ECL, are special cases of the segment predicate defined above. Thus, we re-prove in an entirely different and uniform manner, all the results of [5, 18, 59]

In contrast to previous proofs, the technical essence of the new proofs is to define appropriate codes and to list decode them. These two tasks are independent of the one-way function in question and depend only on the predicate. The only consideration given to the one-way function is for devising a way to access the corrupted codeword (step 3 in the methodology). A task which turns out to be very simple in all the cases considered. We stress that the proofs obtained here are completely different than the previous ones. In particular, the proofs do not require to use the binary gcd algorithm used in previous proofs of the hardcore predicates for RSA [5, 14, 51], nor the square root extraction over finite fields as in [18].

We present a new proof method for simultaneous security of many bits. For this purpose we generalize the notion of balanced hardcore predicates to unbalanced ones. To prove simultaneous bit security, we will show that any violation of simultaneous bit security implies a predictor for some unbalanced hardcore predicate. Using this method we show that a class of functions called *segment functions* – an extension of segment predicates, are simultaneously secure for $RSA, Rabin, EXP$, and ECL. In particular, this implies simultaneous security of the $O(\log\log N)$ most significant bits for those candidate OWFs [64, 86] .

Finally, the new framework applies to proving Goldreich-Levin hardcore predicate in a natural manner where indeed the appropriate code is the Hadamard code.

For a partial summary of these results, see table 1.1.

Predicate (or function) P	Function f	Code $\mathcal{C}^P = \{C_x\}_x$
$GL(x,r)$	$f(x) \colon \{0,1\}^{\lvert r \rvert} \to \{0,1\}^*$	$\{C_x(i) = GL(x,i)\}_{x \in \{0,1\}^n}$
$msb_N(x)$	$RSA_{N,e}(x) = x^e \bmod N$	$\{C_x(i) = msb_N(x \cdot i \bmod N)\}_{x \in \mathbb{Z}_N^*}$
$msb_{p-1}(x)$	$EXP_{p,g}(x) = g^x \bmod p$	$\{C_x(i) = msb_{p-1}(x \cdot i \bmod p-1)\}_{x \in Z_{p-1}^*}$
$msb_q(x)$	$ECL_{p,Q}(x) = xQ$	$\{C_x(i) = msb_q(x \cdot i \bmod q)\}_{x \in Z_q^*}$
$TriLsb_{p-1}$	$EXP_{p,g}(x) = g^x \bmod p$	$\{C_x(i) = TriLsb_{p-1}(x \cdot i \bmod p-1)\}_{x \in Z_{p-1}^*}$
$Pref_N(x)$	$RSA_{N,e}(x) = x^e \bmod N$	$\{C_x^s(i) = D(RSA_{N,e}(x \cdot i \ \bmod N), s)\}$

Table 1.1: Example of predicates (or a function) and codes. Notations details: $GL(z,r) = (-1)^{\langle z,r \rangle}$; $msb_d(z) = 1$ if $0 \leq z < \frac{d}{2}$, -1 o/w; q denotes the order of Q; Assuming $p-1$ is co-prime to 3, $TriLsb_{p-1}(x) = msb(x/3)$ (where the division is modulo $p-1$); $Pref_N(x_1...x_k) = x_1...x_l$ for $l = \log\log N$, $s \in \{0,1\}^l$, and D is a distinguisher for $Pref_N$.

Definition 1.3 (Segment function)**.** *Let* $\mathcal{H} = \left\{h_N \colon Z_N \to \{0,1\}^{l(N)}\right\}_{N \in I}$ *be a collection of functions. For each* $s \in \{0,1\}^{l(N)}$*, define a predicate* $P_N^{\mathcal{H},s} \colon Z_N \to \{0,1\}$*,* $P_N^{\mathcal{H},s}(x) = 1$ *if* $h_N(x) = s$ *and* 0 *otherwise. We say that* $\mathcal{H}$ *is a collection of* segment functions*, if* $\mathcal{P} = \left\{P_N^{\mathcal{H},s}\right\}_{N \in I, s \in \{0,1\}^{l(N)}}$ *is a collection of segment predicates.*

Theorem 1.4. *Let* $\mathcal{H} = \left\{h_N \colon Z_N \to \{0,1\}^{l(N)}\right\}_N$ *be a collection of segment functions. Then* $\mathcal{H}$ *is hardcore for RSA, Rabin,* EXP *and* ECL*, under the assumption that these are OWFs.*

On Diffie-Hellman Hardcore Predicates

The fundamental difference between modern cryptography and the classical one is the use of *public key cryptosystems*, namely, cryptosystems in which the communicating parties exchange only *public keys* and need not know each others *private keys*. The first public *key exchange protocol* was suggested by Diffie and Hellman in their seminal paper "New Directions in Cryptography" [25]. In the Diffie-Hellman protocol, the communicating parties, Alice and Bob, each choose private keys $g^a \bmod p$ and $g^b \bmod p$, respectively (for p a prime, and g a generator of Z_p^*), and exchange the public key $g^{ab} \bmod p$.

The Diffie-Hellman key exchange protocol is based on the *Diffie-Hellman (DH) function*

$$DH_{g,p}(g^a, g^b) = g^{ab} \bmod p$$

for p a prime, and g a generator of the group Z_p^*; and its security is based on the *Decisional Diffie-Hellman assumption (DDH)*, which says that no PPT algorithm can distinguish with non-negligible advantage between the two distributions of $\langle g^a, g^b, g^{ab}\rangle$ and $\langle g^a, g^b, g^r\rangle$, where a, b, r are chosen uniformly at random from Z_p^* (and all the exponentiation are modulo p). A –possibly weaker– assumption is the *Computational Diffie-Hellman assumption (CDH)*, which says that there is no probabilistic polynomial time (PPT) algorithm that, given p, g, g^a, g^b returns g^{ab} (where, again, all the exponentiation are modulo p).

The Diffie-Hellman function and assumptions have been widely used as fundamental primitives in many cryptographic applications. Nonetheless, some of the most central and essential relevant questions have remained open (for a survey see [20]). One such fundamental problems is finding a deterministic hardcore predicate for the Diffie-Hellman function.

Goldreich and Levin [38] showed that for any one-way function f, given $f(x)$ and a random string $r \in \{0,1\}^{|x|}$, it is hard to predict $\sum x_i r_i \bmod 2$. Thus, any one-way function, including the Diffie-Hellman function, has a "randomized" hardcore predicate.

What about deterministic hardcore predicates? For example, is it hard to decide the msb_p predicate, namely, whether the secret g^{ab} is greater or smaller than $\frac{p}{2}$?

For conjectured one-way functions, such as $EXP(x) = g^x \bmod p$ or $RSA(x) = x^e \bmod n$, the first deterministic hardcore predicates were discovered roughly twenty years ago [5,18], and many other deterministic hardcore predicates were discovered since [16,32,41,42,51,59,86,90]. However, *not even one single deterministic hardcore predicates was found for the Diffie-Hellman secret g^{ab}*. A partial result in this direction is the work of Boneh and Venkatesan [19] showing that it is hard to *compute* the $k = O(\sqrt{\log p})$ most significant bits of g^{ab}, given g^a, g^b.

A fundamental question is whether it is possible to improve the result of [19] in: (1) reducing k to one, and (2) showing that even just *predicting* the most significant bit of g^{ab}, given g^a and g^b, with a non-negligible advantage over a random guess cannot be done by a probabilistic polynomial time algorithm, under the CDH assumption. Namely, can we exhibit a deterministic predicate $P\colon \mathbb{Z}_p \to \{\pm 1\}$ such that the existence of a PPT algorithm B that on inputs g^a and g^b returns $P(g^{ab})$ *with non-negligible advantage over a random guess* contradicts the CDH assumption.

We relate the above question to the complexity of the problem of learning characters with noise (LCN), showing that if LCN with random samples access (rLCN) is in BPP, then every segment predicate is hardcore for the Diffie-Hellman function. Furthermore, we show that the latter is true even if easier versions of LCN (such as: LCN with GP-access, that is, with access to samples $\{(x_i, f(x_i))\}_{i=1}^t$ with x_i's a random geometric progression, or LCN with DH-access, that is, with access to samples $f(g^x/g^{ab})$ for any $x = a'b'$ s.t. $g^{a'}, g^{b'}$ can be efficiently computed given g^a, g^b) are in BPP.

These results can be interpreted in two ways. One could either try to find an efficient algorithm for LCN in one of the above access models, with the goal of proving segment predicates are hardcore for the Diffie-Hellman function. Alternatively, one could interpret these results as evidence to the intractability of LCN under those access models.

Definition 1.5 (invDH)**.** *Assuming the hardness of computing the Diffie-Hellman function yields the following collection of OWFs. Define*

$$invDH = \{invDH_{p,g}(g^a, g^b, g^{ab}) = (g^a, g^b),\ invDH_{p,g}\colon \mathbb{Z}_p^3 \to \mathbb{Z}_p^2\}_{\langle p,g\rangle \in I}$$

for $I = \{\langle p, g\rangle,\ p \text{ prime},\ g \text{ a generator of } \mathbb{Z}_p^*\}$.

Definition 1.6. *The access models that arise in our study of DH functions are defined as follows.*

1. **Random samples access.** *We say that an algorithm A_{rand} provides* random samples access *to a function $w\colon \mathbb{Z}_p \to \{\pm 1\}$ if, given p, g, g^a, g^b, A_{rand} outputs a pair $(s, w(s))$ of entry location s and the value of w on this entry, for s distributed uniformly at random in $\mathbb{Z}_p$.*

2. **GP-access.** *We say that an algorithm A_{GP} provides* Geometric Progression access (GP-access) *to a function $w\colon \mathbb{Z}_p \to \{\pm 1\}$ if, given p, g, g^a, g^b and t integers $k_1, \ldots, k_t$, A_{GP} outputs t pairs $\{(s_j, w(s_j))\}_{j=1}^t$ of entry locations s_j and the values of $w(s_j)$ on those entries, where $s_1 = s^{k_1}, \ldots, s_t = s^{k_t}$ for s distributed uniformly at random in $\mathbb{Z}_p$.*

Theorem 1.7. *If LCN with random samples access or GP-access is in BPP, then every segment predicate is hardcore for invDH, under the CDH assumption.*

Technique: List Decoding by Learning Algorithm

The basic approach we take in the list decoding algorithm we develop is to analyze the *Fourier representation* of codewords and corrupted codewords. In particular, for all the codes we use, most of the weight in the Fourier representation of every codeword is concentrated on a small set of characters. We refer to such codes as *concentrated codes*. Furthermore, we prove in the concentration and agreement lemma 6.46 that if w is close to a legal codeword C_x and $\mathcal{C}$ is a concentrated code, then there exists a character χ_α which has a heavy coefficient in both w and C_x.

The above suggests the following high level description of a list decoding algorithm on input word $w\colon \mathcal{D} \rightarrow \{\pm 1\}$: First, apply a *learning algorithm* to find the set of all characters for which w has heavy Fourier coefficients. Then, apply a *recovery algorithm* on each heavy character χ_α found in the first step, to obtain a list L containing all x's for which χ_α is a heavy character of the codeword C_x.

Turning this into a polynomial-time list decoding algorithm for each code considered requires showing the existence of a learning algorithm which given access to an arbitrary function (the corrupted codeword) learns its heavy Fourier coefficients, as well as a recovery algorithm. Whereas learning and recovery algorithms exist for the Hadamard code, we develop them anew for the new codes defined here. As it turns out only one recovery and learning algorithm suffice for the codes we use. The recovery algorithm is elementary. In contrast, the learning algorithm is interesting on his own right.

Further details on our list decoding approach and the algorithm for learning the heavy Fourier coefficients appear in sections 1.2 and 1.3 respectively.

Other Related Works

Hastad and Naslund [51] showed that the ith bit of x (in its binary representation) is a hardcore predicate for the RSA function. We note that this is different than our result showing that the ith partition bit is a hardcore predicate for the RSA function. It is interesting to study further whether the same techniques can be applied to obtain both sets of results.

Fourier analysis of functions over the Boolean cube $\{0,1\}^n$ has been looked at previously in the context of hardcore predicates in the work of Goldmann *et al* [37].

The literature of hardcore predicates is quite vast and many techniques have been employed throughout the years, which we cannot elaborate on here. The technique of Kaliski [59] for proving hardcore predicates for the discrete logarithm problem in general groups might be another interesting avenue to explore in the context of list decoding for discrete log based functions; it also does not use square root extraction of [18] as well.

1.2 A Study of Multiplication Codes

Error correcting codes encode messages into codewords in a way that allows decoding, *i.e.*, recovery of the original messages even from codewords corrupted by some noise. Error correcting codes were introduced by Shannon [76] and Hamming [50] for application such as communication over noisy channels or storage over noisy devices, and have since found many applications in other areas such as complexity and cryptography.

The performance of error correcting codes is measured by an array of parameters, where the parameters to be emphasized and optimized vary according to the application in mind. Examples of performance goals arising in various applications follow.

- **Efficiency, rate and distance.** In communication applications, classical goals are to have efficient encoding and decoding algorithms satisfying that the encoding has high rate (*i.e.*, it introduces only little redundancy) and the decoding algorithm handles a wide range of noise patterns. Explicit codes constructions achieving good performance in terms of the above parameters followed the works of [76] and [50], examples of such codes include: Reed-Solomon [72] and Reed-Muller [68,71] codes, Low-Density-Parity-Check codes [34], and expander based codes [7,44,78,80].

- **Local list decoding.** Applications in complexity and cryptography sometimes introduce alternative goals. For example, Goldreich and Levin [38] in their work on cryptographic hardcore predicates for any one way function introduce the goal of *locally list decoding*, that is, finding all messages whose codeword is close to a given corrupted codeword while reading only a small number of entries in the corrupted codeword. The need for locality emerge from their use of the Hadamard code –a code with exponential encoding length– while working in an unorthodox input model where corrupted codewords are given by black box access rather than being transmitted. Goldreich and Levin [38] provide a local list decoding algorithm for the Hadamard code, presenting the first efficient (local or non-local) list decoding algorithm. Following [38] and Sudan's [81] list decoding algorithm for the Reed-Solomon codes, Sudan-Trevisan-Vadhan [83] presented a local list decoding algorithm for the Reed-Muller codes.

- **Local decoding/self correcting in constant query complexity.** Other applications in complexity and cryptography [9–12,22] achieve stronger locality for the Hadamard codes and for Reed-Muller based codes, in which: recovering each single message bit is done while reading only a *constant* number of corrupted codeword entries. Explicit treatment of such locally decodable codes appear in [60] and in subsequent works; see a survey in [85]. A related goal is to *locally self correct*, that is, to recover the correct value for a single *codeword entry* while reading only a *constant* number of corrupted codeword entries. The abovementioned locally decodable codes –except those appearing in [12]– are also locally self correctable.

In terms of *algebraic structure*, for the vast majority of error correcting codes, the underlying algebraic structure is a *field*. For example, this is the case in all above codes, and more generally, in all *linear codes* (which by definition are codes whose codewords form a subspace of a vector space over some field), as well as in almost all known non-linear codes. An exception are *group codes* [58, 79] – a class of codes extending the class of linear codes by allowing codewords to form a *subgroup* of a group G^n (rather than a subspace of a vector space over a field).

The performance of linear codes, and, more generally, group codes, depends on properties of the underlying group G. In particular, their alphabet size is at least the size of the smallest (non-trivial) subgroup of G. For example, group codes with G a cyclic group of prime order p, have alphabet size p. Group codes (and linear codes) are therefore of interest primarily when the group G (field $\mathbb{F}$) is of *small order* (or has small order subgroups).

Our Work

In this work we introduce a new class of error correcting codes: *Multiplication codes (MPC)*, and develop decoding algorithms for them, showing they achieve desirable combinatorial and algorithmic properties, including: *binary alphabet*, *constant distance* together with efficient *local list decoding* and *local self correcting* algorithms for codes of exponential encoding length, efficient *decoding in random noise* model for codes of polynomial encoding length, and (non-polynomial time) *decoding in adversarial* noise model for codes of *linear encoding length*.

Our results give the first (asymptotic family of) codes achieving constant rate and distance, while having large groups as their underlying algebraic structure.[1] Likewise, our results give the first (asymptotic families of) codes achieving any of the algorithmic properties of being uniquely decodable, list decodable, locally list decodable or locally self correctable, while having large groups as their underlying algebraic structure.

Our techniques and algorithms are applicable beyond the scope MPC: (1) Our local list decoding algorithm is applicable to any Fourier concentrated and recoverable codes.[2] In particular, our algorithm gives a list decoding algorithm for the *homomorphism codes* over any finite abelian group and into the complex numbers. (2) We provide a *soft local error reduction algorithm* for ABNNR codes [7] concatenated with binary codes. This algorithm offers an alternative to Forney's GMD decoding approach for those concatenated codes.

In the following we first define the class of MPC codes. Next, we present the algorithmic and combinatorial results we achieve for MPC codes. We then elaborate

[1]By *large groups* we refer to groups having no (non-trivial) small subgroups, where "small" is, say, constant size, or size logarithmic in information rate. For example, the additive group $\mathbb{Z}_N$ of integers modulo N for $N = pq$ a product of two large primes $p, q = \Omega(\sqrt{N})$ is a large group with respect to the message space $\mathbb{Z}_N^*$ (which has information rate $\log N - o(1)$).

[2]A code is *concentrated* if –when identifying codeword with complex functions over a finite abelian group– its codewords can be approximated by a sparse Fourier representation; a code is *recoverable* if a codeword can be efficiently recognized when given one of its significant Fourier coefficients.

on some applications of our algorithms beyond the scope of MPC. Finally, we mention new techniques we developed for our algorithms.

The Class of Multiplication Codes

Historically, we first defined our MPC codes in the context of our study of cryptographic hardcore predicates for number theoretic one-way functions [3]. We defined there [3] codes $\mathcal{C}^P$ encoding messages $m \in \mathbb{Z}_N$ by values $P(x)$ for $P\colon \mathbb{Z}_N \to \{\pm 1\}$ a Boolean predicate. Specifically, the codeword encoding m is:

$$C(m) = (P(m \cdot 1), P(m \cdot 2), \ldots, P(m \cdot N))$$

where $m \cdot i$ is multiplication modulo N. We gave there [3] a local list decoding algorithm for such codes $\mathcal{C}^P$, provided P is "Fourier well concentrated" (namely, most of the energy of its Fourier transform is concentrated on a small set of significant Fourier coefficients α with $gcd(\alpha, N) \leq poly(\log N)$). This algorithm was inspired by the Goldreich-Levin [38] local list decoding algorithm for the Hadamard codes.

Going beyond the context of cryptographic hardcore predicates, we extend the above definition in two ways. First, to include *codes of good rate*, we extend the definition of MPC to allow restrictions of the above codewords to a subset $S = s_1, \ldots, s_n \subseteq \mathbb{Z}_N$ of their entries. For example, taking S to be a random subset of $\mathbb{Z}_N$ of size $O(\log N)$ the resulting code $\mathcal{C}^{P,const}$ has codewords

$$C^{const}(m) = (P(m \cdot s_1), P(m \cdot s_2), \ldots, P(m \cdot s_n))$$

For this example, we show that for a good choice of the predicate P, the code $\mathcal{C}^{P,const}$ has constant rate and distance.

Furthermore, to include also codes whose *underlying algebraic structure is any abelian group* we further extend the definition of MPC as follows:

Definition 1.8 (Multiplication codes (MPC)). *For any abelian group G, an alphabet controlling function $P\colon \mathbb{C} \to \mathbb{C}$, and a set $S = \{s_1, \ldots, s_n\} \subseteq G$ of indexes to codeword entries,*[3] *we denote by (G, P, S) the MPC code that encode messages $m \in G$ by codewords*

$$C(m) = (P(\chi_m(s_1)), P(\chi_m(s_2)), \ldots, P(\chi_m(s_n)))$$

where $\chi_m\colon G \to \mathbb{C}$ is the homomorphism corresponding to m.[4]

We use the terminology "*MPC code for G*", when we want to emphasize that G is the underlying group. Similarly, when addressing asymptotic families of MPC codes with growing underlying groups G_N, we use the terminology "*MPC code for $\{G_N\}_N$*".

[3]More generally, we also consider MPC where $S \subseteq G \times \ldots \times G$.

[4]The homomorphism $\chi_m\colon G \to \mathbb{C}$ corresponding to m is defined as follows. For $G = \mathbb{Z}_N$, $\chi_m(s) = e^{i2\pi ms/N}$. For $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, $\chi_{m_1,\ldots,m_k}(s_1, \ldots, s_k) = e^{i2\pi \sum_{j=1}^k m_j s_j / N_j}$. In general, for G a multiplicative group with a generating set $\{g_1, \ldots, g_k\}$ where g_j has order N_j, $\chi_{\prod_{j=1}^k g_j^{m_j}}(\prod_{j=1}^k g_j^{s_j}) = e^{i2\pi \sum_{j=1}^k m_j s_j / N_j}$.

MPC codes as a class of codes extends the class of (abelian) group codes and the class of linear codes.

Remark 1.9. *In retrospect, any MPC code can be thought of as being defined by a group code $\mathcal{C} \subseteq \Sigma_1^n$ together with an alphabet reduction function $P\colon \Sigma_1 \to \Sigma_2$ mapping codewords $(C_1, \ldots, C_n) \in \Sigma_1^n$ of the group code $\mathcal{C}$ into codewords $(P(C_1), \ldots, P(C_n)) \in \Sigma_2^n$ of the MPC. The challenge is to find group codes $\mathcal{C}$ and alphabet reduction functions P that result in an MPC codes achieving good combinatorial and algorithmic properties.*

The alphabet reduction method method we present here is useful even for very large alphabet. *For example, we use it to reduce the alphabet of homomorphism codes (i.e., an alphabet which is as large as the message space). This is in contrast to the well known* codes concatenation *alphabet reduction method, which is useful only in small alphabet settings.*[5]

Multiplication Codes Achieving Good Properties

Our main result is presenting three (asymptotic families of) MPC codes for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ all achieving binary alphabet and constant distance[6] and the following algorithmic properties and encoding length: (1) Codes of encoding length N which are *locally list decodable* and *locally self correctable.* (2) Codes of *encoding length polynomial in* $\log N$, which are decodable in *random noise model* in time polynomial in $\log N$, and are decodable in *adversarial noise model* in time $N^{2\varepsilon}poly(\log N)$ (for ε the fraction of flipped bits). (3) Codes of *constant rate and distance.*

We use the notation $(k, n, d)_q$-code referring to codes of message space size 2^k, encoding length n, normalized Hamming distance d, and alphabet size q.

Theorem 1.10 (MPC codes for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$). *We present three (asymptotic families of) MPC codes for groups $\mathbb{Z}_N$ of growing sizes N:*

1. **Codes with local algorithms***:* $(\Theta(\log N), N, \Theta(1))_2$*-codes, which are efficiently* locally list decodable *and* locally self correctable*. The local list decoding algorithm we present has query complexity and running time poly*$(\log N)$*. The local self correcting algorithm we present has query complexity* $\Theta(1)$ *and running time poly*$(\log N)$*. The input to the local self correcting algorithm is restricted to entries $s \in \mathbb{Z}_N^*$. Both algorithms are randomized.*[7]

[5] In *codes concatenation*, alphabet Σ_1 is reduced to a smaller alphabet Σ_2 by encoding each symbol $\sigma \in \Sigma_1$ using a code of alphabet Σ_2. This method is of interest only when the alphabet Σ_1 is considerably smaller than the message space size (otherwise finding a good code for the message space Σ_1 is as hard as finding a good code for the original message space). That is, concatenation is only useful when the alphabet in not too large to begin with.

[6] The code distance is the minimum relative Hamming distance between any two of its codewords, where the relative Hamming distance is the fraction of entries on which they differ.

[7] In the local list decoding algorithm success probability is taken only over the random coins of the algorithm, namely, it is independent of the input; success probability $1 - \rho$ is achieved in time *poly*$(\log N, \ln(1/\rho))$. In the local self correcting algorithm we present in this chapter, success probability is taken both over the input and over the random coins of the algorithm; success probability

2. **Polynomial rate codes with efficient decoding***:* $(\Theta(\log N), poly(\log N), \Theta(1))_2$-*codes, which are efficiently decodable in the random noise model. The decoding algorithm we present runs in time* $poly(\log N)$*. Furthermore, these codes are decodable in adversarial noise model in time* $N^{2\varepsilon} poly(\log N)$ *for* ε *the fraction of flipped bits.*

3. **Codes of constant rate & distance***:* $(\Theta(\log N), \Theta(\log N), \Theta(1))_2$-*codes.*

Remark 1.11. *For linear codes, local decodability is implied by any local self correcting algorithm robust against the changes of basis (because there is a basis change transforming the code to* systematic*, where message bits appears as part of the codeword). In contrast, for* non-*linear codes —as are MPC— a systematic representation does not necessarily exist, and a relation between local self correcting and local decoding is not known.*

Extending the above results to other groups, we present MPC codes for any finite abelian group with a constant size generating set. These codes achieve parameters as in the above theorem, albeit with alphabet size $2^{O(k)}$ for $k = \Theta(1)$ the generating set size. The algorithms we present receive a description of the underlying group by its generators and their orders as part of the input.

Theorem 1.12 (MPC for groups of small generating sets)**.** *For any finite abelian* G *with a generating set of size* $k = O(1)$*, there exists MPC codes for* G *achieving performance as in the above theorem, albeit with alphabet size* $2^{O(k)}$*.*

Remark 1.13. *When all generators have the same order, the codes we present have alphabet size* 2^k*. In particular, for cyclic groups, the alphabet is binary, i.e., of size* 2*.*

Locally List Decoding Fourier Concentrated Codes

Our local list decoding algorithm is applicable to any code $\mathcal{C}$ whose codewords are Fourier concentrated and recoverable (when identified with complex functions over a finite abelian group). Namely, our algorithm is applicable to codes $\mathcal{C} \subseteq \{C : G \to \mathbb{C}\}$, G a finite abelian group, satisfying that: (1) codewords can be approximated by a sparse Fourier representation[8] and (2) codewords can be efficiently recognized when given one of their significant Fourier coefficients.

Theorem 1.14. *Let* $\mathcal{C}$ *be a Fourier concentrated and recoverable code, denote its* ℓ_2-*distance by* $d = \min_{C,C' \in \mathcal{C}} \|C - C'\|_2^2$*. There is an algorithm that given a corrupted codeword* w *and a distance parameter* $\varepsilon \in [0, d)$ *returns a short list containing all*

$1 - \rho$ is achieved in query complexity $poly(1/\rho)$ and running time $poly(\log N, 1/\rho)$. This latter algorithm can be improved to achieve success probability depending *only* on the random coins of the algorithm. Such improvements are out of scope for this thesis.

[8] A sufficient condition is that for any codeword C, the sum of its Fourier coefficients is of size $\sum_\alpha \left|\widehat{C}(\alpha)\right| \leq poly \log |G|$.

codewords at ℓ_2-distance $\|C - w\|_2^2 \leq d - \varepsilon$ from w; and its running time at most poly$(\log |G| / \varepsilon)$.
Remark. *In particular, the output list contains all codewords at normalized Hamming distance from w of at most $\frac{d}{4} - \frac{\varepsilon}{4}$.*[9]

An immediate example of Fourier concentrated and recoverable codes are the homomorphism codes from a finite abelian group G and into the complex.[10] Thus a corollary of the above is a list decoding algorithm for homomorphism codes w.r. to ℓ_2 distance:

Corollary 1.15. *Let $\mathcal{C}$ be a homomorphism codes over a finite abelian group G and into the complex unit sphere. There is an algorithm that given a corrupted codeword w and a distance parameter $\varepsilon \in [0, 2)$ returns a short list containing all codewords at ℓ_2-distance $\|C - w\|_2^2 \leq 2 - \varepsilon$ from w; and its running time at most poly$(\log |G| / \varepsilon)$.*
Remark. *In particular, the output list contains all codewords at normalized Hamming distance from w of at most $\frac{1}{2} - \frac{\varepsilon}{4}$.*

Remark 1.16. *An equivalent way to think of the SFT algorithm is as a local list decoding algorithm (in the ℓ_2 distance) for the homomorphism codes. This is because a codeword C_m of the homomorphism code is close to a corrupted codeword w in the ℓ_2-distance iff the m-th Fourier coefficient of w is a significant coefficient.*

Soft Error Reduction for Concatenated ABNNR Codes

Alon *et.al.* [7] presented a distance amplification scheme for error correcting codes relying on expansion properties of expander graphs; aka ABNNR codes. Guruswami and Indyk [44] presented an error reduction algorithm for binary codes (aka, the inner code) concatenated with the ABNNR codes (aka, the outer code). The error reduction algorithm of [44] follows Forney's GMD methodology [33] for decoding concatenated codes, where an efficient decoding algorithm is required for the inner code, and an efficient algorithm for error reducing from erasures and errors is required for the outer code.

We present an alternative to Forney's GMD decoding approach that does not require an erasures-and-errors error reduction algorithm for the outer code. Instead we take a *soft decoding approach* where the inner code returns a list of potential codewords along with their distances from the (corresponding portion of the) received word, and the outer code incorporates these lists of distances to find the closest codeword.

The algorithm we present is *local*, namely, each bit of message symbol can be found with a d^2 queries to the codewords for d the degree of the underlying expander graph.

[9]This is due to the bound $\|C_m - w\|_2^2/2^2 \leq \Delta(C_m, w) \leq \|C_m - w\|_2^2/|\Sigma|^2$ for $\Delta(C_m, w) = \Pr[C_m(s) \neq w(s)]$ the normalized Hamming distance, and Σ the code alphabet when interpreted as complex roots of unity of order $|\Sigma|$, that is, $\Sigma = \left\{e^{i\frac{2\pi}{|\Sigma|}t}\right\}_{t=1}^{|\Sigma|}$.

[10]Homomorphism code from (say, an additive group) G into the complex unit sphere is the set of all functions $\chi \colon G \to \mathbb{C}$ satisfying that $\chi(x + y) = \chi(x)\chi(y)$ and $|\chi(x)| = 1$ for all $x \in G$.

Our soft decoding approach gives the same optimal distance performance as in [44], and may be of interest outside the realm of MPC codes, *e.g.*, for decoding concatenated codes where the outer code has no efficient erasures-and-errors decoding algorithm.

New Techniques

Decoding via learning (in query and subset access models). For our decoding algorithms we develop a *decoding via learning approach*, where we identify codewords with complex functions over a finite abelian group (or, restrictions of such functions to a subset of their entries), and decode by first finding the significant Fourier coefficients of the given corrupted codeword, and then mapping the significant Fourier coefficients to the messages of the close codewords.

Finding significant Fourier coefficients: For codes whose codewords are functions over finite abelian groups, we find their significant Fourier coefficients using our SFT algorithm. For codes whose codewords are restrictions of such functions to a subset of the finite abelian group, we develop new algorithms for finding their significant Fourier coefficients. These algorithms find significant Fourier coefficients of a signal when given values of the signal on a (carefully designed) predetermined set of entries and where values are corrupted by (random or adversarial) noise.

Mapping significant Fourier coefficients to messages of close codewords: For linear codes (or, more generally, group codes) the significant Fourier coefficient immediately map to the messages encoded by the close codewords. For the MPC codes that we present, such a map is not always immediate. Nevertheless, we show that such a map exists and can be computed efficiently.

Self correcting via testing. Our local self correcting algorithm is composed of two parts: (1) A reduction from the self correcting problem to the property testing problem of distinguishing between signals with high and low Fourier coefficients in a large interval, and (2) An algorithm for solving the property testing problem.

Both the reduction algorithm and the property testing algorithm make only a constant number of queries to the given corrupted codeword.

Other Related Works

The idea of *list decoding* —that is, finding the list of all codewords close to the given corrupted codeword— was proposed in the 1950's by Elias [28] and Wozencraft [89] for dealing with situations where the noise is too great to allow unique decoding. Efficient list decoding algorithms followed many years later, starting with the Goldreich and Levin [38] list decoding algorithm for the Hadamard code, which is a code of exponentially small rate (*i.e.*, codeword length is exponential in message length). A few years later, Sudan [81] presented the first polynomial time list decoding algorithm for codes of polynomial rate: the Reed-Solomon codes. In following years more list decoding algorithms were presented, including improved list decoding algorithm for Reed-Solomon codes [48], and polynomial time list decoding algorithms for: Reed-Muller codes [83], Algebraic Geometric codes [48, 77], Chinese Remainder codes [39], certain concatenated codes [49], graph based codes [45], "XOR lemma based" codes [84] (codes defined there), and folded Reed-Solomon codes [47]. Following our work [3], Grigorescu *et.al.* [43] presented a list decoding algorithm for homomorphism codes whose domain and range are both any finite abelian group, their algorithm correct errors up to large relative Hamming distances.

The list decoding algorithms of Goldreich-Levin [38], Sudan-Trevisan-Vadhan [83], Trevisan [84] and Grigorescu *et.al.* [43] are *local* list decoding algorithms, where the query complexity is polynomial in $\log N$ for N the codeword length and in the distance parameter in [38, 43, 83], and it is of the order of $\sqrt{N}$ and exponential in the distance parameter in [84].

1.3 Learning Characters with Noise

Fourier Analysis is among the most widely used tools in electrical engineering and computer science, employed in numerous applications taken from versatile areas such as polynomial multiplication, filtering noise in audio or visual signals, lossy compression, and sequence retrieval.

In the context of theoretical computer science, Fourier analysis of functions over the Boolean cube $\{0,1\}^n$ has found many applications in areas including cryptography, complexity theory, computational learning theory and property testing; a few examples follow.

> **In cryptography.** Goldreich and Levin [38] in their work on hard-core predicates for any one-way function, develop as a tool an algorithm for finding significant Fourier coefficients of boolean functions over the Boolean cube $\{0,1\}^n$ in time polynomial in n (when given query access to the function).[11]
>
> **In complexity.** Linial Mansour and Nisan [63] utilize Fourier analysis to prove a lower bound on constant depth circuits AC^0 via showing that an AC^0 Boolean function has almost all of its Fourier spectrum on the low-order Fourier coefficients. Later, Hastad [53] introduces the use of Fourier analysis to the field of Hardness-of-Approximation, and gives many tight inapproximability results for well known problems. Dinur and Safra [26] followed by Khot-Regev [61] further incorporate Fourier analysis to strengthen the previously known hardness-of-approximation results for the Vertex-Cover problem.
>
> **In computational learning.** Kushilevitz and Mansour [62] present an algorithm for learning decision trees, which employs as a central tool the algorithm of [38] for finding significant Fourier coefficients. Jackson's [57] further employ the algorithm of [38] for learning DNF formulas. Both algorithms are in the membership queries model.

[11]The use of Fourier analysis is implicit in the paper of Goldreich and Levin [38], and was made explicit by Kushilevitz and Mansour [62].

The problem of Learning Parity with Noise (LPN) is also (implicitly) related to Fourier analysis: In LPN, the goal is to find $s \in \{0,1\}^n$, when given samples $(a, \sum_{j=1}^n s_i a_i + \eta)$ for $a \in \{0,1\}^n$ chosen at random and $\eta \in \{0,1\}$ a noise value satisfying that $\eta = 0$ whp. In Fourier language, the goal is to find the close character χ_s,[12] when given samples $(a, \chi_s(a) \cdot \eta)$ for noise $\eta \in \{\pm 1\}$ with $\eta = 1$ whp. Different distributions of a were considered in literature, *e.g.*, uniform distribution, or distribution generated by a random walk on the hypercube [21].

Without noise, LPN is tractable using Gauss Elimination. With noise, LPN is believed to be intractable in the worst case, and consequently also in the average case due to the random self reducibility of LPN. The fastest algorithm for LPN, given by Blum-Kalai-Wasserman [15], runs in time $2^{O(n/\log n)}$. The conjectured intractability of LPN (or, equivalently, the conjectured intractability of decoding random linear codes) is employed in cryptographic protocols, including the McEliece cryptosystem [66] and signature scheme [24], the Hopper-Blum authentication protocol [55] and subsequent works regarding efficient authentication protocols [88].

In property testing. Fourier analysis was employed in several works, including: the analysis of Bellare *et.al.* [13] to the Blum Luby and Rubinfeld [17] linearity test, Hastad's [52] long code test, and Fischer *et.al.* [31] juntas test.

In the context of approximation theory, algorithms extending Goldreich-Levin [38] algorithm to find significant Fourier coefficients of more general function classes were developed. Mansour [65] gave such an algorithm for complex functions over groups $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ provided that $N_1, \ldots, N_k$ are *powers of two*. Gilbert *et.al.* [35] gave such an algorithm for real functions over $\mathbb{Z}_N$ for any positive integer N.

In recent years, Fourier analysis of functions over other domains has proved useful also in theoretical computer science: Fourier analysis of functions over real vector spaces $\mathbb{R}^n$ was introduced in the context of lattice based cryptosystems by Regev [73], and further employed in complexity analysis of lattice problems [1, 67]; Fourier analysis of functions over additive groups $\mathbb{Z}_N$ of integers modulo N was introduced in the context of cryptographic hardcore predicates in a joint work with Goldwasser and Safra [3];[13] and Fourier analysis of functions over small finite fields $\mathbb{Z}_p^n$ for prime $p \leq poly(n)$ was employed by Regev [74], again in the context of lattice based cryptosystems.

We observe that one computational problem underlies many of the aforementioned uses of Fourier analysis. We name the problem *Learning Characters with Noise (LCN)*:

LCN (informally). Given a specification of a function f, the goal is to find all Fourier characters χ that are close to f (in some distance measure).

[12]The characters over the Boolean cube $\{0,1\}^n$ are $\{\chi_s \colon \{0,1\}^n \to \{\pm 1\}\}_{s \in \{0,1\}^n}$ defined by $\chi_s(a) = (-1)^{\sum_{j=1}^n s_i a_i}$.

[13]The work of Akavia Goldwasser and Safra [3] is presented as a part of this dissertation.

For example, in the Goldreich-Levin [38] algorithm for finding significant Fourier coefficient, the input is boolean functions over the Boolean cube $f\colon \{0,1\}^n \to \{0,1\}$ specified by query access, and the goal is to find all characters χ close to f in the Hamming distance.

In the problem of Learning Parity with Noise, the input is taken from the class of functions f obtained by adding noise to a character $\chi\colon \{0,1\}^n \to \{0,1\}$; the input is specified by random samples $(a, f(a))$; and the goal is to find the character χ closest to f in Hamming distance.

In Mansour's [65] algorithm for sparse approximation of polynomials, the input is a complex function f over a field of size 2^n (or more generally, a product of such fields) specified by query access, and the goal is to find all characters highly correlated with f, *i.e.*, characters close to f in the ℓ_2-distance.

Our Work

As part of this thesis we provide a systematic study of the problem of *Learning Characters with Noise (LCN)*. We develop algorithms for the LCN problem for (1) different function classes, *e.g.*, periodic functions over the integers and functions over general finite abelian groups; (2) different function specification models, *e.g.*, query access to the function, and random samples of the function; and (3) different approximation metrics, *e.g.*, the Euclidean and Hamming metrics.

In LCN, given a specification of a function f, the goal is to find all Fourier characters χ that are close to f (in some distance measure). We define LCN in broad settings, where the input function f may be any complex function over an abelian group, the specification of f may be given by any algorithm M that upon request returns a sample of f, possibly depending on values sent with the request (aka, *M-access*), and the distance d may be arbitrarily defined.

Different instantiations of LCN are defined by a class $\mathcal{F}$ from which the input functions are drawn, the access model M and the distance measure d.

Definition 1.17 (LCN)**.** *The input to LCN is M-access to $f \in \mathcal{F}$ and a threshold $\tau > 0$. The goal is to find all characters*[14] *χ s.t. $d(f,\chi) \geq \tau$.*

We study the following instantiations of LCN, differing on their access models, their considered function classes, and their distance measures:

- **qLCN.** Given a description[15] of a finite abelian group G, $\tau > 0$ and query access to a function $f\colon G \to \mathbb{C}$, the goal is to find all τ-significant Fourier coefficients of f.[16]

  By *query access* we mean that the learning algorithm can ask and receive the value $f(x)$ for every $x \in G$ in unit time.

[14]For $f\colon G \to \mathbb{C}$ the characters are all homomorphisms χ over G and into the complex unit sphere $\mathbb{C}$. For example, for $G = \mathbb{Z}_N$ the characters are all functions $\chi_\alpha(x) = e^{i2\pi\alpha x/N}$, $\alpha \in \mathbb{Z}_N$.

[15]A description of G is given by its generators and their orders.

[16]The τ-significant Fourier coefficients of $f\colon G \to \mathbb{C}$ are all α s.t. $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$ for $\widehat{f}(\alpha) = \frac{1}{|G|}\sum_{x\in G} f(x)\chi_\alpha(x)$ the coefficient of the characters χ_α in the Fourier representation of f.

- **rLCN.** Given a description of a finite abelian group G, $\tau > 0$, and random samples access to a function $f\colon G \to \mathbb{C}$, the goal is to find the characters χ s.t. $d(\chi, f) < \tau$.

  By *random samples access* we mean that the learning algorithm can ask and receive samples $(x, f(x))$ for x drawn uniformly at random from G in unit time.

  We consider various distance measures d including: *Hamming distance*; ℓ_∞ *distance*; ℓ_2 *distance*; *combined primes* and *distinct primes* distances which are measures we define to explore relations between LCN for functions over product groups (say, $\{0,1\}^n$) and LCN for functions over cyclic groups (say, $\mathbb{Z}_N$); and randomized variants of the former models.

- **Interval-LCN.** Given N, $\tau > 0$ and interval-access to a function $f\colon \mathbb{Z}_N \to \mathbb{C}$, the goal is to find all τ-significant Fourier coefficients of f.

  By *interval-access* we mean that the learning algorithm can ask and receive samples $\{(x_i, f(x_i))\}_{i=1}^m$ that are correlated by having all x_i's drawn from the same (randomly chosen) interval.

- ***GP*-LCN.** Given a description of a finite abelian group G, $\tau > 0$ and GP-access to a function $f\colon G \to \mathbb{C}$, the goal is to find all τ-significant Fourier coefficients of f.

  By *GP-access* we mean that the learning algorithm can ask and receive samples $\{(x_i, f(x_i))\}_{i=1}^m$ s.t. $x_1, \ldots, x_m$ form a random geometric progression in the group G.[17]

- ***Q*-LCN.** Given a description of a finite abelian group G, $\tau > 0$ and Q-access to a function $f\colon G \to \mathbb{C}$, the goal is to find all τ-significant Fourier coefficients of f.

  By *Q-access* we mean that the learning algorithm can ask and receive the value $f(x)$ for every $x \in Q$ in unit time, where $Q \subseteq G$ is a predetermined subset of G.

Terminology. For $\mathcal{G}$ a set of groups (usually, infinite), we abbreviate by saying *"LCN for $\mathcal{G}$"* when addressing LCN with function class restricted to the class of complex functions over groups $G \in \mathcal{G}$. *E.g.*, qLCN for finite abelian groups is the qLCN problem when the input function f is taken from class of complex functions over finite abelian groups. Likewise, rLCN for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ is the rLCN problem when the input functions f is taken from class of complex functions over groups $\mathbb{Z}_N$ for some positive integers N.

We focus on algorithms for LCN with running time polynomial in the bit representation length of elements in the domain of f, that is, polynomial in $\log N$ for N

[17]More generally, we consider also GP-access where on input integers $k_1, k_2, ..., k_m$, the access algorithm M outputs samples $(r, \{f(r^{k_i})\}_{i=1}^m)$ for $r \in G$ distributed uniformly at random and where the power operation is computed in the group G; and its running times is polynomial in m and $\log|G|$.

the size of the domain of f; aka *polynomial algorithms*. We say that an instantiation of LCN is *tractable* if there exists a polynomial algorithm for it, saying it is *intractable* otherwise.

In the following we present our results for LCN, first addressing LCN in query access model, then – LCN in random samples access model, and finally – LCN in intermediate access models of interval access, GP-access and subset access. Table 1.2 summarizes our algorithmic results for LCN, presenting in rows 1-11 our polynomial time algorithms, and in rows 12-13 our non-polynomial time algorithms.

	Input Model	Distance Measure	Function Class	Running Time
1	query access	ℓ_2	$\{f\colon G \to \mathbb{C}\}_{G\in\mathcal{G}}$	$poly(\log\|G\|)$
2		no noise		
3		Hamming		
4	random samples	ℓ_∞	$\{f\colon \mathbb{Z}_N \to \mathbb{C}\}_{N\in\mathbb{N}}$	$poly(\log N)$
5		random Hamming		
6		random ℓ_∞		
7	random samples	combined primes	$\{f\colon \mathbb{Z}_N \to \mathbb{C}\}_{N\in\mathcal{I}_1}$	$poly(\log N)$
8	random samples	distinct primes	$\{f\colon \mathbb{Z}_N \to \mathbb{C}\}_{N\in\mathcal{I}_2}$	$poly(\log N)$
9	interval access	ℓ_2	$\{f\colon \mathbb{Z}_N \to \mathbb{C}\}_{N\in\mathbb{N}}$	$poly(\log N)$
10	subset access	ℓ_2	$\mathcal{F} \subseteq \{f\colon G \to \mathbb{C}\}_{G\in\mathcal{G}}$	$poly(\log\|G\|, \log\|\mathcal{F}\|)$
11	subset access	ℓ_2	well spread faulty $\mathcal{F}$, $\mathcal{F} \subseteq \{f\colon G \to \mathbb{C}\}_{G\in\mathcal{G}}$	$poly(\log\|G\|, \log\|\mathcal{F}\|)$
12	subset access	ℓ_2	δ-spread faulty $\mathcal{F}$, $\mathcal{F} \subseteq \{f\colon \mathbb{Z}_N \to \mathbb{C}\}_{N\in\mathbb{N}}$	$N^{2\delta}poly(\log N, \log\|\mathcal{F}\|)$
13	random samples	ℓ_2	$\{f\colon \mathbb{Z}_N \to \mathbb{C}\}_{N\in\mathbb{N}}$	$N^{1-o(1)}$

Table 1.2: **Algorithms for LCN.** This table summarizes our algorithms for various instantiations of LCN. We assume without loss of generality that the input functions are bounded in range $\|f\|_\infty \in [-1,1]$ and in energy $\|f\|_2 \leq 1$. Each row describes one algorithm, specifying the input model, distance measure and function class to which it is applicable, and its running time dependency on domain size (omitting dependency on noise parameter). Rows 2-6 all share the same input model, function class and running time, and vary only on the distance measure. In rows 1,10-11, $\mathcal{G}$ is the set of all finite abelian groups. In row 7, $\mathcal{I}_1$ is the set of all positive integers $N = \prod_{i=1}^t p_i$ that are a product of $t > 1$ distinct primes s.t. $\sum_{i=1}^t \frac{N}{p_i}$ is co-prime to N and is efficiently computable. In row 8, $\mathcal{I}_2$ is the set of all positive integers $N = \prod_{i=1}^t p_i$ that are a product of $t > 1$ distinct primes s.t. $p_1, \ldots, p_t$ can be efficiently found given N. In rows 10-12, the subset access is to a subset of size $poly(\log N, \log|\mathcal{F}|)$ for N the size of the domain of f. In rows 11-12, the given function is a faulty version of a function in $\mathcal{F}$, where faults are "well spread" in row 11, and "δ-spread" in row 12 for "well-spread" and "δ-spread", in particular, for Boolean functions, the Binary Symmetric Channel results in well spread faults (whp), whereas adversarial channel flipping $O(\delta)$-fraction of the bits results in δ-spread faults.

LCN in Query Access Model (qLCN)

In qLCN, given a description of a finite abelian group G (by its generators and their orders), $\tau > 0$ and query access to a function $f\colon G \to \mathbb{C}$; the goal is to find all τ-significant Fourier coefficients of f.

A naive solution for this problem would compute the entire Fourier transform of f, and then output only the significant coefficients; thus running in time $O(N \log N)$ for $N = |G|$. This running time is exponential in the binary representation length $\log N$ of elements in G, which is the parameter we measure our running time against.

The *SFT algorithm* presented in this thesis solves qLCN over any finite abelian group G in running time $\widetilde{\Theta}(\log N)$ for $N = |G|$.

Theorem 1.18 (SFT algorithm)**.** *There is a probabilistic algorithm solving qLCN in running time polynomial in* $\log |G|$, $1/\tau$ *and* $\|f\|_\infty$.
Remark. *In particular, for* $f\colon \mathbb{Z}_N \to \{\pm 1\}$*, the running time is* $\widetilde{\Theta}\left(\frac{\log N}{\tau^{5.5}}\right)$.

We remark that our SFT algorithm is different than the algorithm of [35] even when restricted to functions over $\mathbb{Z}_N$, improving on the latter in achieving $\widetilde{\Theta}(\log N)$ rather than $poly(\log N)$ dependency on N.

LCN in Random Samples Access Model (rLCN)

In rLCN for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$, given $N \in \mathbb{N}$, $\tau > 0$, and random samples access to a function $f\colon \mathbb{Z}_N \to \mathbb{C}$, the goal is to find the characters χ of $\mathbb{Z}_N$ s.t. $d(\chi, f) < \tau$ (for d some distance measure). We consider various distance measures d.

On the positive side, we show that rLCN for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ is *tractable* in Hamming distance for any $\tau \in [0, 1)$, and in the ℓ_∞ distance for τ poly-logarithmically related to the group size.

Theorem 1.19 (Tractability in Hamming distance)**.** *rLCN over* $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ *with Hamming distance measure and* $\tau < 1 - \frac{1}{poly \log N}$ *is tractable.*
Remark. *When* $\tau < \frac{1}{2}$*, the algorithm can return the* unique *closest character in running time polynomial in* $\log N$ *and* $1/(\frac{1}{2} - \tau)$*. In general, the algorithm returns the* list *of* $O(1/\tau)$ *closest characters in running time polynomial in* $\log N$ *and* $1/(1-\tau)$.

Theorem 1.20 (Tractability in ℓ_∞ distance)**.** *rLCN over* $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ *with* ℓ_∞ *distance measure and* $\tau < \sin(\frac{2\pi}{N} poly \log N)$ *is tractable.*

Remark 1.21. *Similar results hold for the random Hamming and random* ℓ_∞ *distances.*

Furthermore, we investigate the correspondence between rLCN for *cyclic groups* and rLCN for *product groups*. For this purpose we define noise models –*distinct primes* and *combined primes*– that simulate a product group structure for cyclic groups $\mathbb{Z}_N$ with $N = p_1 p_2 \dots p_t$ a product of distinct primes. This is by exploiting the isomorphism of such groups $\mathbb{Z}_N$ with product groups $\mathbb{Z}_{p_1} \times \dots \times \mathbb{Z}_{p_t}$. We then explore the complexity of rLCN in those noise models, showing that its tractability/intractability depends on properties of the factorization of N.

On the negative side, we *conjecture* that rLCN for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ is *intractable* in the ℓ_2 distance. Moreover, we conjecture it is still intractable even when the input functions are restricted to be *Boolean* (aka, *"Boolean ℓ_2 distance"*). We bring supporting evidence to these conjectures relating them to applications in cryptography and coding theory. Specifically, we show that if these conjectures are false, then we can (1) decode random (non-linear) binary codes of constant rate and distance, and (2) present the first deterministic hardcore predicate for the Diffie-Hellman function.

Theorem 1.22. *Let $\mathcal{C}^{const}$ be the random code from section 1.2. If there is an algorithm solving rLCN over $\mathbb{Z}_N$ with Boolean ℓ_2 distance in poly$(\log N)$ time, using $O(\log N)$ samples, and with error probability poly$(\frac{1}{N})$, then there is an algorithm for decoding $\mathcal{C}^{const}$ in time poly$(\log N)$ and with success probability at least $2/3$.*

Theorem 1.23. *If rLCN over $\{\mathbb{Z}_p\}_{prime\ p}$ with Boolean ℓ_2 distance is tractable, then every segment predicate*[18] *is hardcore for the Diffie-Hellman function.*

In addition, we show that in the ℓ_2 and the Boolean ℓ_2 distances, rLCN for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ is *random self reducible.* That is, with those distances, if no algorithm solves rLCN on the worst case, then no algorithm solves rLCN on the average case (within the same input sizes).

Theorem 1.24 (Random self reducibility)**.** *rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ with ℓ_2 distance is random self reducible. Moreover, it is random self reducible even when restricted to the class of Boolean functions.*

Finally, we present a (non-polynomial time) algorithm for rLCN in the ℓ_2 and the Boolean ℓ_2 distances, running in time $N^{1-O(1/(\log\log N)^2)}$ for N the size of the domain of f. This gives a slight improvement over the $O(N\log N)$ running time of the naive algorithm.[19]

Theorem 1.25 (algorithm for rLCN)**.** *There is a probabilistic algorithm solving rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ with ℓ_2 distance and with $\tau = O(1)$ sufficiently small in time $N^{1-poly(1/\log\log N)}$.*

LCN in Intermediate Access Models (Interval-LCN, GP-LCN, Q-LCN)

The random samples access and query access models represent two extremes with respect to the control the learning algorithm has on the samples it gets: Total control — in the query access, where in turn we have an efficient algorithms for LCN in ℓ_2 distance measure; No control — in the random samples access, where in turn we conjecture LCN in ℓ_2 distance measure to be intractable.

[18]Segment predicates are defined in Definition 7.8, and include for example the most significant bit predicate.

[19]The naive algorithm is an algorithm measuring correlation of $O(\log N)$ random samples with each character of $\mathbb{Z}_N$, outputting the character with highest correlation.

In applications, achieving query access is not always feasible; whereas, partial control over the samples is sometimes still available. This leads to defining intermediate access models where the algorithm has partial, albeit incomplete, control over the samples it sees. We cosider several examples of such intermediate access models arising in applications.

In the *interval access model*, the learning algorithm can ask and receive samples $\{(x_i, f(x_i))\}_{i=1}^m$ that are correlated by having all x_i's drawn from the same (randomly chosen) interval. We show that interval-LCN (*i.e.*, LCN for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ with interval access) is tractable. An application of this result is our (non-polynomial time) algorithm for solving LCN in random samples access model in the ℓ_2 distance measure.

Theorem 1.26. *Interval-LCN is in BPP.*

In the *GP-access model*, the learning algorithm can ask and receive samples $\{(x_i, f(x_i))\}_{i=1}^m$ s.t. $x_1, \dots, x_m$ form a random geometric progression in the group G. We show that tractability of GP-LCN (*i.e.*, LCN with GP-access) is related to the problem of proving bit security of the Diffie-Hellman (DH) function. In particular, we show that either GP-LCN is intractable, or several bits of DH are as secure as DH.

Theorem 1.27. *If GP-LCN is tractable, then every segment predicate (as defined in Definition 7.8) is hardcore for the Diffie-Hellman function.*

In the *subset access model*, the learning algorithm can ask and receive the value $f(x)$ for every $x \in Q$ in unit time, where $Q \subseteq G$ is a predetermined subset of G. We show that for any family $\mathcal{F}$ of functions there are subsets Q of size polynomial in $\log|G|, \log|\mathcal{F}|$ such that Q-LCN (*i.e.*, LCN with Q-access, that is, subset access to the subset Q) is tractable for any function f in $\mathcal{F}$. We use extensions of this algorithm to noisy setting as a component in our algorithms for decoding polynomial rate codes discussed in section 1.2.

Definition 1.28 (($\mathcal{Q}, \mathcal{F}$)-LCN)**.** *Let $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathcal{I}}$ be a family of functions $\mathcal{F}_N \subseteq \{f\colon G_N \to \mathbb{C}\}$ over finite abelian groups G_N, and let $\mathcal{Q} = \{Q_{N,\tau}\}_{N\in\mathcal{I},\tau\in\mathbb{R}^+}$ a family of subsets $Q_{N,\tau} \subseteq G_N$. The input to $(\mathcal{Q}, \mathcal{F})$-LCN is a description of a finite abelian group G_N, $\tau > 0$ and $Q_{N,\tau}$-access to a function $f \in \mathcal{F}_N$; the goal is to find all the τ-significant Fourier coefficients of f.*[20]

Theorem 1.29. *For every family $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathcal{I}}$ of functions $\mathcal{F}_N \subseteq \{f\colon G_N \to \mathbb{C}\}$ over finite abelian groups G_N, there exists a family of polynomial size subsets $\mathcal{Q} = \{Q_{N,\tau} \subseteq G_N\}_{N\in\mathcal{I},\tau\in\mathbb{R}^+}$ s.t. $(\mathcal{Q}, \mathcal{F})$-LCN is in P.*

Remark 1.30. *Explicit construction of the sets $Q_{N,\tau}$ are possible, using explicit construction of small biased sets to G_N. Moreover, such explicit constructions of size*

[20]We say that $\alpha \in G_N$ is a τ-significant Fourier coefficients of $f\colon G_N \to \mathbb{C}$ if the α Fourier coefficient of f is of weight at least $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$, where $\left|\widehat{f}(\alpha)\right| = \langle f, \chi_\alpha\rangle$ is the coefficient of the character χ_α in the Fourier representation of f.

poly$(\log|G|, M)$ *allow solving* Q*-LCN for any function* f *satisfying* $\sum_\alpha \left|\widehat{f}(\alpha)\right| \leq M$. *(This is similar to Alon and Mansour [6] de-randomization of Mansour's [65] algorithm solving qLCN for functions over fields* $\mathbb{Z}_{2^n}$ *(or products of such fields).)*

Other Related Works

Feldman *et.al.* [30] consider another aspect of average and worst case relation, showing equivalence of Learning Parity with Noise (LPN) in average case *noise*, that is, random noise, and in worst case noise. The corresponding question regarding rLCN is yet to be determined.

We observe that intermediate access models were (implicitly) studied for LPN in the subset access model. Specifically, we observe that every linear error correcting code defines a subset $Q \subseteq \{0,1\}^n$ such LPN with query access to Q is tractable when restricted to functions f that are the corrupted codewords. Since there are linear codes of constant rate, then there are linear size subsets $|Q| = O(n)$ such that Q-LPN with is tractable (with restricted input functions).

1.4 Algorithms for Data Intensive Applications

Our modern times are characterized by information explosion incurring a need for faster and faster algorithms. Even algorithms classically regarded efficient —such as the Fast Fourier Transform (FFT) algorithm with its $\Theta(N \log N)$ complexity— are often too slow for data-intensive applications, and linear or even sub-linear algorithms are imperative. Despite the vast variety of fields and applications where algorithmic challenges arise, some basic algorithmic building blocks emerge in many of the existing algorithmic solutions. Accelerating such building blocks can therefore accelerate many existing algorithms. One of these recurring building blocks is the Fast Fourier Transform (FFT) algorithm.

Computing the Fourier transform of a signal of length N may be done in time $\Theta(N \log N)$ using the Fast Fourier Transform (FFT) algorithm. This time bound clearly cannot be improved below $\Theta(N)$, because the output itself is of length N. Nonetheless, it turns out that in many applications it suffices to find only the *significant Fourier coefficients*, *i.e.*, Fourier coefficients occupying, say, at least 1% of the energy of the signal.

Our SFT algorithm may offer a great efficiency improvement over the FFT algorithm for applications where it suffices to deal only with the significant Fourier coefficients. In such applications, replacing the FFT building block with the SFT algorithm accelerates the $\Theta(N \log N)$ complexity in each application of the FFT algorithm to $\widetilde{\Theta}(\log N)$ complexity. A few examples of such applications follow.

- **Lossy Compression.** A central component in several transform compression methods (*e.g.*, JPEG) is to first apply Fourier (or Cosine) transform to the signal, and then discard many of its coefficients. To accelerate such algorithms —instead of computing the entire Fourier (or Cosine) transform— the SFT algorithm can be used to directly approximate only the significant Fourier coefficients. Such an accelerated algorithm achieves compression guarantee as good as the original algorithm (and possibly better), but with running time improved to $\widetilde{\Theta}(\log N)$ in place of the former $\Theta(N \log N)$.

- **Data Retrieval.** Fourier transform is used in several data retrieval algorithms. As an example, consider *similarity search*, namely, a database search where the goal is to retrieve all sequences within a certain Euclidean distance from a given query. For large databases, a similarity search might be infeasible, and instead, it is preferable to conduct the comparison over shorter sequences, called *fingerprints*. The fingerprints may be defined in various ways, for example, in [29, 70] the fingerprint of a sequence is its *first few Fourier coefficient*.

  Fingerprinting sequences by their first few Fourier coefficient is a very general method, and may, in principal, be applied to versatile types of databases, such as databases of textual data, DNA sequences, images or audio signals. However, this fingerprinting method is *worthwhile* only when most of the mass of the sequences is concentrated on their first few coefficients.

  Utilizing our SFT algorithm, we can fingerprint each sequence by its *few most significant Fourier coefficients* instead of taking the first ones. This alternative fingerprinting method might turn worthwhile for applications where the first Fourier coefficients are not significant, and thus the existing algorithm cannot be used.

- **SETI.** When using Fourier transform to analyze the radio telescope data in search for Extraterrestrial Intelligence (SETI), one is not interested in the specific values of the Fourier transform, but rather is only interested in distinguishing between the case that *the signal is a random noise* from the case that *the signal is originated from some Extraterrestrial Intelligence*. One way to distinguish random noise from a potentially meaningful signal is to find whether the Fourier transform of the signal has some *peak*, namely, some entry on which a noticeable fraction of the signal's mass is concentrated. Using our SFT algorithm to check whether the signal has a peak may be done in time $\widetilde{\Theta}(\log N)$, instead of the $\Theta(N \log N)$ complexity when using FFT to compute the entire Fourier transform.

- **Function Approximation.** A straightforward application of our SFT algorithm is for finding sparse Fourier approximation for function f over finite abelian groups, when given query access f.

Related Works

Lossy compression using fast algorithms for qLCN has caught a wide interest under the name of *compressed sensing* [27], *e.g.*, in [23]. Applications of fast algorithms for qLCN to dimensionality reduction were suggested by Gilbert *et.al.* [36]. More examples of function classes that can be efficiently approximated using our SFT algorithm are presented by Atici and Rocco [8].

1.5 Conclusions & Thesis Organization

In this work we initiated the systematic study of the *Learning Characters with Noise (LCN)* problem, presenting in particular efficient algorithms for several of its instantiations. We then employ these algorithms in cryptography and in error correcting codes.

In cryptography we introduce a unifying framework for proving that a Boolean predicate is hardcore for a one-way function, and apply it to a broad family of predicates and candidate one-way functions.

In *coding theory*, we introduce a new class of error correcting codes that we name: *Multiplication codes (MPC).* We develop decoding algorithms for MPC codes, showing they achieve desirable combinatorial and algorithmic properties. MPC are unique in particular in achieving properties as above while having no underlying field structure, but rather a large group structure.

An additional contribution of our work is in introducing the use –in cryptographic and coding theoretic context– of Fourier analysis of complex functions over the group $\mathbb{Z}_N$ of integers modulo N for large values of N, and demonstrating its usefulness.

Acknowledgments. Many of the results presented in this thesis are due to joint work with Goldwasser and Safra [3]. This includes the SFT algorithm, the first definition of the MPC codes and their local list decoding algorithm, the decoding via learning approach for concentrated and recoverable codes, and all our results in cryptography except the ones on Diffie-Hellman function.

Addressing LCN in various access models, its relation to hardcore predicates for the Diffie-Hellman application, as well as uses of the SFT algorithm in data intensive application are due to a joint work with Goldwasser [2].

The soft local error reduction algorithm for ABNNR codes concatenated with binary codes is due to a joint work with Venkatesan [4].

Thesis Organization The organization of the rest of this thesis is as follows. In chapter 2 we present some preliminary notations, terminology and facts. In chapter 3 we present our results on the LCN problem in the query access model. In chapter 4 we present our results on the LCN problem in the random samples access model. In chapter 5 we present our results on the LCN problem in intermediate access models. In chapter 6 we present our results on the MPC codes. In chapter 7 we present our results on cryptographic hardcore predicates.

Chapter 2

Preliminaries

In this chapter we survey preliminary definitions, notations and facts that are used throughout this work.

2.1 Notations and Terminology

We use the terms *non-negligible* and *negligible* to address polynomial and smaller than polynomial sizes: We say that $\rho(\cdot)$ is *non-negligible* if there exists a constant $c \geq 0$ and an integer k_c such that $\rho(k) > k^{-c}$ for all $k \geq k_c$. Another way to think of it is $\rho(k) = k^{-\Theta(1)}$. We say that a function $\nu(\cdot)$ is *negligible* if for every constant $c \geq 0$ there exists an integer k_c such that $\nu(k) < k^{-c}$ for all $k \geq k_c$. Another way to think of it is $\nu(k) = k^{-\omega(1)}$.

We denote the *prejudice* of Boolean functions f toward their majority and minority values by maj_f and minor_f: For every Boolean function $f\colon D \to \{\pm 1\}$ over a domain D, denote by

$$\mathsf{maj}_f \overset{def}{=} \max_{b \in \{\pm 1\}} \Pr_{x \in D}[f(x) = b]$$

the *prejudice* of f toward its majority value, and by $\mathsf{minor}_f \overset{def}{=} 1 - \mathsf{maj}_f$ the prejudice of f toward its minority value. For a family of Boolean functions $\mathcal{F} = \{f_N \colon D_N \to \{\pm 1\}\}$, denote

$$\mathsf{maj}_{\mathcal{F}} \overset{def}{=} \limsup_{f \in \mathcal{F}} \mathsf{minor}_{f_N}$$

and $\mathsf{minor}_{\mathcal{F}} \overset{def}{=} \liminf_{f \in \mathcal{F}} \mathsf{minor}_{f_N}$.

Balanced Boolean functions f have no more than a negligible prejudice toward their majority value, namely, both minor_f and maj_f are roughly $\frac{1}{2}$: We say that $\mathcal{F}$ is *balanced* if for every $f_N \in \mathcal{F}$, $\left|\mathsf{minor}_{f_N} - \mathsf{maj}_{f_N}\right| < \nu(\log N)$ for ν a negligible function.

To simplify expression of running time and sample complexity, we sometimes use $\widetilde{\Theta}$ notation which overlooks quantities poly-logarithmic in appearing arguments, that is, $\widetilde{\Theta}(x) = \Theta\left(x \cdot (\log x)^{\Theta(1)}\right)$.

We denote the set of natural numbers by $\mathbb{N}$, the set of integers by $\mathbb{Z}$, the set of real number by $\mathbb{R}$, the set of complex numbers by $\mathbb{C}$, the set of complex number of unit magnitude by $\mathbb{T}$. We denote by $\mathbb{Z}^+$ and $\mathbb{R}^+$ the set of positive integers and positive reals, respectively.

We denote by $\mathbb{Z}_N$ the additive group of integers modulo N. For every element $x \in \mathbb{Z}_N$, we denote by

$$\mathsf{abs}(x) \stackrel{def}{=} \min\{x, N-x\}$$

the "distance" of x from zero in the group $\mathbb{Z}_N$.

2.2 Fourier Transform over Finite Abelian Groups

We specify terminology and facts regarding finite abelian groups and the Fourier transform of functions over finite abelian groups.

Abelian groups. An *abelian (or commutative) group* is a pair $(G, *)$ for G a set of elements, and $*$ an operation on the elements in G satisfying the following. G is closed under the operation $*$, that is, $\forall x, y \in G$, $x * y \in G$; the operation $*$ is associative, that is, $\forall x, y, z \in G$, $x * (y * z) = (x * y) * z$; the operation $*$ is commutative, that is, $\forall x, y \in G$, $x * y = y * x$; and G has a neutral element e with respect to the operation $*$, that is, $\forall x \in G$, $x * e = e * x = x$. A group is *finite* if its set of elements G is finite. In our notation, we often omit the operation $*$ and denote groups by their set of elements G.

By the Fundamental Theorem of Abelian Groups every finite abelian group G is isomorphic to a direct product of cyclic groups, that is, $G \cong \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2} \times \ldots \times \mathbb{Z}_{N_k}$ for some positive integers $N_1, \ldots, N_k$. Moreover, there are elements $g_1, \ldots, g_k \in G$ of orders $N_1, \ldots, N_k$ respectively s.t. each $x \in G$ can be uniquely represented as a product

$$x = g_1^{x_1} \ldots g_k^{x_k}$$

for $(x_1, \ldots, x_k) \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$. A description of the group G can therefore be given by the elements $g_1, \ldots, g_k$ and their orders $N_1, \ldots, N_k$. In the following chapters when we say *a description of G by its generators and their orders*, we refer to generators forming a generating set with *unique* representation.

Definition 2.1 (Group description). *Let $g_1, \ldots, g_k \in G$ and $N_1, \ldots, N_k$ their orders*[1]. *For a multiplicative groups G, we say that $\{(g_i, N_i)\}_{i=1}^k$ is a* description of G *if every element $x \in G$ can be uniquely represented as a product $x = \prod_{i=1}^k g_i^{x_i}$ for $x_1, \ldots, x_k \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$.*

Similarly, for an additive groups G, we say that $\{(g_i, N_i)\}_{i=1}^k$ is a description of G *if every element $x \in G$ can be uniquely represented as a sum $x = \sum_{i=1}^k x_i \cdot g_i$ for $x_1, \ldots, x_k \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$.*

We denote the description of G by $\mathsf{Info}(G)$.

[1]For a multiplicative group G, N is the order of $g \in G$ if $g^N = 1$ and $\forall t < N$, $g^t \neq 1$. Similarly, for an additive group G, N is the order of $g \in G$ if $N \cdot g = 0$ and $\forall t < N$, $t \cdot g \neq 0$.

For example, the description of the Boolean cube $G = \{0,1\}^n$ is k pairs $(1,2)$; the description of the additive group $G = \mathbb{Z}_N$ of integers modulo N is $(1,N)$; the description of the product group $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ is $\{(1,N_i)\}_{i=1}^k$.

In the following we focus on product groups $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, definitions for other finite abelian follow from their isomorphism with such product groups.

Characters. Denote by $\omega_N \stackrel{def}{=} e^{i\frac{2\pi}{N}}$ the primitive root of unity of order N, that is, ω_N is a complex number of unit magnitude s.t. $\omega_N^N = 1$ and $\omega_N^k \neq 1$ for any integer $1 \leq k < N$. The *characters* of G are all homomorphisms from G to the complex number of unit magnitude. That is,

$$\text{Characters of } G = \{\chi\colon G \to \mathbb{T} \mid \chi(x+y) = \chi(x)\cdot\chi(y)\ \forall x,y \in G\}$$

The characters of G with coordinate-wise addition form a group which is isomorphic to G. In particular, the number of characters of G is equal to the number of elements in G. We index the characters of G by elements G. For $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ and each $(\alpha_1,\ldots,\alpha_k) \in G$, the character $\chi_{(\alpha_1,\ldots,\alpha_k)}\colon G \to \mathbb{T}$ is defined by

$$\chi_{(\alpha_1,\ldots,\alpha_k)}(x_1,\ldots,x_k) = \omega_{N_1}^{\alpha_1 x_1} \ldots \omega_{N_k}^{\alpha_1 x_k}$$

Similarly, characters of arbitrary finite abelian groups G are defined according to the group $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ isomorphic to G. That is, for G a finite abelian group of description $\{(g_i,N_i)\}_{i=1}^k$, its characters are $\left\{\chi_{g_1^{\alpha_1}\ldots g_k^{\alpha_k}} : G \to \mathbb{T}\right\}_{(\alpha_1,\ldots,\alpha_k)\in\mathbb{Z}_{N_1}\times\ldots\times\mathbb{Z}_{N_k}}$ defined by

$$\chi_{g_1^{\alpha_1}\ldots g_k^{\alpha_k}}(g_1^{x_1}\ldots g_k^{x_k}) = \omega_{N_1}^{\alpha_1 x_1} \ldots \omega_{N_k}^{\alpha_1 x_k}$$

Inner product, norms and convolution. For any functions $f,g\colon G \to \mathbb{C}$, their *inner product* (or, correlation) is

$$\langle f,g\rangle \stackrel{def}{=} \frac{1}{|G|}\sum_{x\in G} f(x)\overline{g(x)}$$

The ℓ_2-norm (or, energy) of f is

$$\|f\|_2 \stackrel{def}{=} \sqrt{\langle f,f\rangle}$$

and its ℓ_∞-norm (or, maximal amplitude) is

$$\|f\|_\infty \stackrel{def}{=} \max\{\,|f(x)| \mid x \in G\}$$

The *convolution* of f and g is the function $f * g\colon G \to \mathbb{C}$ defined by

$$f * g(x) \stackrel{def}{=} \frac{1}{|G|}\sum_{y\in G} f(y)\overline{g(x-y)}$$

Fourier transform, significant coefficients and their approximation. The Fourier transform of functions $f\colon G \to \mathbb{C}$ is the function $\widehat{f}\colon G \to \mathbb{C}$ that measures the correlation of f with the characters in G, that is, for each $\alpha \in G$,

$$\widehat{f}(\alpha) \overset{def}{=} \langle f, \chi_\alpha \rangle$$

For any $\alpha \in G$, $val_\alpha \in \mathbb{C}$ and $\tau, \varepsilon \in \mathbb{R}^+$, we say that α is a *τ-significant Fourier coefficient* iff

$$\left|\widehat{f}(\alpha)\right|^2 \geq \tau$$

we say that val_α is an *ε-approximation for* $\widehat{f}(\alpha)$ iff

$$\left|val_\alpha - \widehat{f}(\alpha)\right| < \varepsilon$$

We denote the set of τ-significant Fourier coefficients of f by $\mathsf{Heavy}_\tau(f)$, that is,

$$\mathsf{Heavy}_\tau(f) \overset{def}{=} \left\{\alpha \in G \mid |\langle f, \chi_\alpha \rangle|^2 \geq \tau\right\}$$

The Fourier transform has several useful properties: (1) The Fourier transform of the convolution of two functions is equal to the coordinate-wise product of their Fourier transforms. (2) The inner product of two functions is equal (up to normalization) to the inner product of their Fourier transforms. (3) The Fourier transform of a phase shifted function is a translation of its its Fourier transform prior to the phase shift. (4) The Fourier transform of a function whose inputs are multiplicatively shifted by an element a is equal to a multiplicative shift by an elements a^{-1} of its Fourier transform prior to the multiplicative shift. These properties are stated formally in the following theorem.

Theorem 2.2 (Properties of Fourier Transform)**.** *Let $f, g\colon \mathbb{Z}_N \to \mathbb{C}$.*

1. **Convolution-Multiplication Duality:** $\forall \alpha \in \mathbb{Z}_N$, $\widehat{f * g}(\alpha) = \widehat{f}(\alpha) \cdot \widehat{f}(\alpha)$

2. **Plancherel Theorem or Parseval Identity:** $\langle f, g \rangle = \sum_{\alpha \in \mathbb{Z}_N} \left|\widehat{f}(\alpha)\right|^2 |\widehat{g}(\alpha)|^2$

3. **Translation:** *If* $\exists \alpha_0 \in \mathbb{Z}_N$ *s.t.* $\forall x \in \mathbb{Z}_N$, $g(x) = f(x)\chi_{-\alpha_0}(x)$, *then* $\forall \alpha \in \mathbb{Z}_N$, $\widehat{g}(\alpha) = \widehat{f}(\alpha - \alpha_0)$ *(where subtraction is modulo N)*

4. **Dilation:** *If* $\exists a \in \mathbb{Z}_N^*$ *s.t.* $\forall x \in \mathbb{Z}_N$, $g(x) = f(ax)$, *then* $\forall \alpha \in \mathbb{Z}_N$, $\widehat{g}(\alpha) = \widehat{f}(\alpha/a)$ *(where multiplication and division are modulo N)*

2.3 Error Correcting Codes

Error correcting codes encode messages in a way that allows recovery of the original message even in the presence of noise.

Codes and binary codes. A *code* $\mathcal{C} = \{C_x\}_{x \in D} \subseteq \Sigma^n$ is a set of length n vectors over an alphabet Σ. The vectors $C_x \in \mathcal{C}$ are called *codewords*. Each codeword C_x encodes a message x taken from a *message space* denoted by D. A *binary code* has alphabet $\Sigma = \{\pm 1\}$.

We identify the alphabet Σ with a set of complex numbers, and restrict our attention to codes with codewords of bounded ℓ_2-norm, $\|C_x\|_2^2 \leq 1 \; \forall x \in D$. We often think of codewords as complex valued functions mapping entry location to codeword value.

Rate and distance. The *rate* of the code measures the amount of information it contains per codeword bit, *i.e.*,

$$r(\mathcal{C}) = \frac{\log |D|}{\log |\Sigma^n|}$$

The *Hamming distance* of two vectors $f, g \in \Sigma^n$ is the number of entries on which they differ,

$$Hamming(f, g) = |\{\, i \in [n] \mid f_i \neq g_i \}|$$

The *relative Hamming distance* of f, g is the Hamming distance normalized by the vectors length,

$$\Delta(f, g) = \frac{1}{n} Hamming(f, g)$$

The *relative code distance* is the minimum relative Hamming distance over all codewords, *i.e.*,

$$\Delta(\mathcal{C}) = \min_{C_x, C_y \in \mathcal{C}} \Delta(C_x, C_y)$$

For any $w \in \Sigma^n$ and $\eta \geq 0$, the *Hamming ball of radius* η *around* w in $\mathcal{C}$ is the set of codewords at relative Hamming distance at most η from w, that is,

$$\mathcal{B}_{\mathcal{C},Hamming}(w, \eta) = \{\, C_x \in \mathcal{C} \mid \Delta(C_x, w) < \eta \}$$

The ℓ_2 *ball of radius* η *around* w in $\mathcal{C}$ is the set of codewords at normalized ℓ_2 distance at most η from w, that is,

$$\mathcal{B}_{\mathcal{C},\ell_2}(w, \eta) = \left\{\, C_x \in \mathcal{C} \mid \|C_x - w\|_2^2 < \eta \right\}$$

When the distance measure and the code are clear from context we sometimes omit them and denote the ball by $\mathcal{B}(w, \eta)$.

Notation. We say that a code $\mathcal{C} = \{C_x\}_{x \in D} \subseteq \Sigma^n$ is a

$$(n, k, \delta)_q\text{-code}$$

if the message space is of information length $\log |D| = k$, the encoding length is n,the code distance is $\Delta(\mathcal{C}) \geq \delta$, and the alphabet is of size $|\Sigma| = q$.

Noise Models. The *Binary Symmetric Channel* noise model of parameter ε, denoted BSC_ε, corrupts codewords $C_x \in \mathcal{C}$ by flipping each bit independently at random with probability ε. That is, on input a codeword $C_x \in \{\pm1\}^n$ the BSC_ε channel outputs a vector $C_x + \eta \in \{\pm1\}^n$ for $\eta \in \{-2,0,2\}^n$ a noise pattern s.t. $\forall i \in [n]$ the value on entry i is chosen independently at random to be $\eta_i = 0$ with probability $1-\varepsilon$ and $\eta_i = -2C_x(1)$ otherwise.

Adversarial noise model of parameter ε corrupts codewords $C_x \in \mathcal{C}$ by changing them to the worst case vector w in the Hamming ball of radius ε around C_x. We also consider *adversarial noise of parameter ε w.r. to the ℓ_2 distance*, where the channel outputs the worst case vector w in the ℓ_2 ball or radius ε around the input codeword C_x.

Infinite families of codes. We consider *infinite families of codes* $\mathcal{C} = \{\mathcal{C}_N\}_{N\in\mathcal{I}}$ for $\mathcal{I}$ is an infinite set of indexes, and for $\mathcal{C}_N = \{C_{N,x}\}_{x\in D_N} \subseteq \Sigma_N^{n_N}$ codes with codewords of length n_N and alphabet Σ_N that encode messages from the message space D_N. The *rate* of the code family $\mathcal{C}$ is

$$r(\mathcal{C}) = \liminf_{N\in\mathcal{I}} r(\mathcal{C}_N)$$

The *relative distance* of the code family $\mathcal{C}$ is

$$\Delta(\mathcal{C}) = \lim \inf_{N\in\mathcal{I}} \Delta(\mathcal{C}_N)$$

For n, k, δ, q functions from $\mathcal{I}$ to the positive integers, we say that the code family $\mathcal{C}$ is a $(n,k,\delta)_q$-code, if for all $N \in \mathcal{I}$, $\mathcal{C}_N$ is a $(n_N, k_N, \delta_N)_{q_N}$-code.

Codes and groups. In our work we study codes $\mathcal{C} = \{C_x\}_{x\in D} \subseteq \Sigma^n$ corresponding to finite abelian groups G. We define codes corresponding to groups G as follows: The messages are taken from the group G, that is, the message space D is a subset of G. The codewords entries are identified with a subset $S \subseteq G$ of size $|S| = n$. The codewords are identified with complex valued functions over S, $C_x\colon S \to \Sigma$, mapping entry location $s \in S$ to the codeword value $C_x(s)$ on that entry.

List Decoding. For binary codes $\mathcal{C} = \{C_x\colon G \to \{\pm1\}\}_{x\in D}$ with balanced codewords, we say that $\mathcal{C}$ is *list decodable*, if there is an algorithm that, given a corrupted codeword $w\colon G \to \{\pm1\}$ and a distance parameter $\varepsilon \in [0, \frac{1}{2})$, returns a list

$$L \supseteq Ball_{\mathcal{C},Hamming}(w, \varepsilon)$$

containing all messages x' whose codeword $C_{x'}$ is in the ball of radius ε around w; and its running time is polynomial in $\log|G|$ and $1/(\frac{1}{2}-\varepsilon)$. We remark that the running time bound in particular implies that the list size $|L|$ is polynomial in $\log|G|$ and $1/(\frac{1}{2}-\varepsilon)$.

For unbalanced binary codes the definition is similar, only replacing $\frac{1}{2}$ with minor_w. That is, $\varepsilon \in [0, \mathsf{minor}_w)$ and the running time is polynomial in $\log|G|$ and $1/(\mathsf{minor}_w -$

ε).

Equivalently, one can think of a list decoding algorithm as returning a list L containing all codewords $C_{x'}$ highly correlation with w, that is,

$$L \supseteq \{ x' \mid \langle w, C_{x'} \rangle \geq \widehat{w}(0) + 2\varepsilon \}$$

(This is by observing that for binary codes $\langle f, g \rangle = 1 - 2\Delta(f,g)$ and $\frac{1}{2} - \frac{1}{2} \left| \widehat{w}(0) \right| = \mathsf{minor}_w$.)

For non binary codes $\mathcal{C} = \{ C_x \colon G \to \Sigma \}_{x \in D}$, we say $\mathcal{C}$ is *list decodable* if there is an algorithm that, given a corrupted codeword $w \colon G \to \Sigma$ and a distance parameter $\varepsilon > 0$, returns a list L containing all codewords that are highly correlated with w, that is,

$$L \supseteq \{ x' \mid \langle w, C_{x'} \rangle \geq \widehat{w}(0) + \varepsilon \}$$

We remark that, in particular, when C_x, w accept values in the unit complex sphere $\mathbb{T}$, the list decoding algorithm returns all codewords in the Hamming ball of radius $\varepsilon' = \frac{1}{2} - \frac{1}{2}(\widehat{w}(0) + \varepsilon)$ around w.[2]

2.4 One-Way Functions and Hardcore Predicates

We study the cryptographic primitive called *one-way functions*, focusing on *hardcore predicates* for one-way functions. Let us formally define those primitives.

A one-way function is a function that is easy to compute but hard to invert. There are two equivalent definitions of (strong) one-way functions one may work with. The first is a *single* function defined over an *infinite* domain which is asymptotically hard to invert with high probability (where the probability is taken over all strings of the same length). The second definition is suitable for number theoretic OWFs candidates, and therefore, we use this formulation in this work.

Definition 2.3 (OWF). *We say that $\mathcal{F} = \{ f_i \colon \mathcal{D}_i \to R_i \}_{i \in I}$ is a* collection of one-way functions (OWFs) *(where I is an infinite set of indexes, $\mathcal{D}_i$ and R_i are finite), if (1) one can efficiently sample $i \in I \cap \{0,1\}^k$ (2) $\forall i \in I$, one can efficiently sample $x \in \mathcal{D}_i$ (3) $\forall i \in I, x \in \mathcal{D}_i$, one can efficiently compute $f_i(x)$ and (4) $\forall i \in I$, f_i is (strongly) hard to invert, namely, for every PPT algorithm A there exists a negligible function ν_A such that*

$$\Pr[f_i(z) = y \colon y = f_i(x), z = A(i,y)] < \nu_A(k)$$

(here the probability is taken over random choices of $i \in I \cap \{0,1\}^k$, $x \in \mathcal{D}_i$, and over the coin tosses of A).

[2] We show that the correlation ball $B_c = \{ C_x \mid \langle C_x, w \rangle \geq \varepsilon + \widehat{w}(0) \}$ of radius $\widehat{w}(0) + \varepsilon$ around w contains the Hamming ball $B_h = Ball_{\mathcal{C},Hamming}(w, \varepsilon')$ of radius $\varepsilon' = \frac{1}{2}(1 - \widehat{w}(0) - \varepsilon)$ around w. Let $C_x \in B_h$, we show that $C_x \in B_c$. Denote $\eta = w - C_x$. We express the correlation of w, C_x in terms of η: $\langle w, C_x \rangle = \langle C_x, C_x \rangle + \langle \eta, C_x \rangle$. The first summand $\langle C_x, C_x \rangle$ is equal to 1, since C_x accepts values on the unit sphere. The second summand is $\langle \eta, C_x \rangle$ is at least $-2\varepsilon'$, because $\eta(i)$ is 0 on at least $1 - \varepsilon'$ fraction of its entries for C_x in Hamming ball of radius ε', and for non zero entries $\eta(i) C_x(i) \geq -2$ for C_x, w accepting values on the unit sphere. For $\varepsilon' = \frac{1}{2}(1 - \widehat{w}(0) - \varepsilon)$ we get that $\langle w, C_x \rangle \geq 1 - 2\varepsilon' = \widehat{w}(0) + \varepsilon$, implying that $C_x \in B_c$.

A hardcore predicate captures the hardness in inverting the one-way function by singling out a Boolean predicate whose value is as hard to predict as it is hard to invert the one-way function. A Boolean function P is a *hardcore predicate for a function* f if (1) it is easy to compute $P(x)$ given x, but (2) it is hard to predict the value of $P(x)$ given $f(x)$ with any non-negligible advantage over a random guess. We are interested in Boolean predicates, usually considering predicates taking ± 1 values rather than $0, 1$ values.

Definition 2.4 (Predicts)**.** *Let $\mathcal{P} = \{P_i \colon \mathcal{D}_i \to \{\pm 1\}\}_{i \in I}$ be a collections of Boolean predicates. Denote by $\mathsf{maj}_{P_i} = \max_{b \in \{\pm 1\}} \Pr_x[P_i(x) = b]$ the* prejudice *of P toward its majority value. We say that an* algorithm B predicts P_i from f_i *if $\exists$ a non-negligible function ρ s.t.*

$$\Pr[B(i, f_i(x)) = P_i(x)] \geq \mathsf{maj}_{P_i} + \rho(k)$$

where the probability is taken over the random coins tosses of B and choices of $x \in \mathcal{D}_i \cap \{0,1\}^k$.

In particular, for balanced predicates $\mathcal{P}$,[3] *we say that an* algorithm B predicts P_i from f_i *if $\exists$ a non-negligible function ρ s.t.*

$$\Pr[B(i, f_i(x)) = P_i(x)] \geq \frac{1}{2} + \rho(k)$$

where the probability is taken over the random coins tosses of B and choices of $x \in \mathcal{D}_i \cap \{0,1\}^k$.

Definition 2.5 (Hardcore predicates)**.** *Let $\mathcal{P} = \{P_i \colon \mathcal{D}_i \to \{\pm 1\}\}_{i \in I}$ be a collection of Boolean predicates, and $\mathcal{F} = \{f_i \colon \mathcal{D}_i \to R_i\}_{i \in I}$ a collection of one-way functions. We say that $\mathcal{P}$ is* hardcore for $\mathcal{F}$ *if (1) it there is a PPT algorithm that computes $P_i(x)$ given i and x, but (2) there is no PPT algorithm that predicts P_i from f_i.*

In what follows we fix a one-way function $f_i \in \mathcal{F}$ and a predicate $P_i \in \mathcal{P}$, $i \in I$, and present an efficient reduction from inverting f_i with non-negligible probability to predicting $P_i(x)$ given $f_i(x)$. This will suffice for showing that $\mathcal{P}$ is a hardcore predicate for $\mathcal{F}$. In this case, we abuse notations and say that P_i is hardcore of f_i.

For a full definition of one-way collection of functions and hardcore predicatess for collections see [40].

2.5 Tail Inequalities in Probability

Tail inequalities bound the probability that a random variable accept values far from its expectations. Two such inequalities are stated below.

Theorem 2.6 (Chernoff/Hoeffding Bound [54])**.** *Suppose that $X_1, \ldots, X_t$ are independent random variables of expectation μ_i each. Further suppose that they assume*

[3] $\mathcal{P} = \{P_i \colon \mathcal{D}_i \to \{\pm 1\}\}_{i \in I}$ is a collection of *balanced predicates* if $\forall i \in I$, $|\Pr_x[P_i(x) = 1] - \Pr_x[P_i(x) = -1]| < \nu(k)$ for some negligible function ν.

values only from $\{0, \ldots, M\}$*. Then, for any* η*,*

$$\Pr\left[\frac{1}{t}\sum_{i=1}^{t} X_i - \frac{1}{t}\sum_{i=1}^{t}\mu_i \geq \eta\right] \leq 2 \cdot exp\left(-\frac{2t\eta^2}{M^2}\right)$$

Theorem 2.7 (Chebyshev)**.** *Let* X *be a random variable with expectation* μ *and variance* σ^2*. Then for any real number* $k > 0$*,*

$$\Pr\left[|X - \mu| > k\sigma\right] < \frac{1}{k^2}$$

Chapter 3

Learning Characters with Noise in Query Access Model

In this chapter we present our SFT algorithm for finding significant coefficients in the Fourier transform of complex function over arbitrary finite abelian groups.

3.1 Introduction

In this chapter we study the problem of Learning Characters with Noise with *query access* model and ℓ_2 *distance* measure. In this problem, given query access to a function f, the goal is to find all characters that are close to f in the ℓ_2 norm.[1] Equivalently, the goal is to find all α s.t. the α Fourier coefficient of f, $\left|\widehat{f}(\alpha)\right|^2$, is large. We denote this problem by *qLCN*.

Definition 3.1 (Query access)**.** *Let $f\colon G \to \mathbb{C}$. We say that an algorithm is given* query access *to f, if it can request and receive the value $f(x)$ for any $x \in G$ in unit time.*

Definition 3.2 (qLCN)**.** *The input to qLCN is a description*[2] *of a finite abelian group G, a noise parameter $\tau \in \mathbb{R}^+$ and query access to a function $f\colon G \to \mathbb{C}$; the goal is to find all τ-significant Fourier coefficients of f.*[3]

A *naive* solution to qLCN is to compute the Fourier transform of f, and output only the significant coefficients. This naive algorithm runs in time $O(N \log N)$ for $N = |G|$, which is *exponential* in the the binary representation length $n = \log N$ of elements in G – the parameter we measure our running time against.

[1]We focus in this chapter on the ℓ_2 distance measure. The problem of Learning Characters with Noise in other distance measures are studied in chapter 4. For several distance measures (*e.g.*, the Hamming distance) we show there that LCN is tractable already in the more challenging access model of *random samples access*. This in particular implies tractability in the query access model.

[2]A description of G is given by its generators and their orders as specified in Definition 2.1.

[3]We say that $\alpha \in G$ is a *τ-significant* Fourier coefficient of f iff $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$.

We present an efficient algorithm —that is, with running time polynomial in $n = \log N$— that solves qLCN over any abelian group G. We name our algorithm the *SFT Algorithm*.

Theorem 3.3 (SFT algorithm)**.** *There is a probabilistic algorithm solving qLCN in running time polynomial in* $\log|G|$, $1/\tau$ *and* $\|f\|_\infty$.
Remark. *In particular, for* $f\colon \mathbb{Z}_N \to \{\pm 1\}$*, the running time is* $\widetilde{\Theta}\left(\frac{\log N}{\tau^{5.5}}\right)$.

The SFT algorithm follows a line of works solving qLCN in restricted function classes. Goldreich and Levin [38] gave an algorithm for Boolean functions over $\{0,1\}^n$. In the context of approximation theory, Mansour [65] gave an algorithm for complex functions over $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ provided that $N_1, \ldots, N_k$ are *powers of two.* Gilbert *et.al.* [35] gave an algorithm for real functions over $\mathbb{Z}_N$ for any positive integer N.

Our SFT algorithm is different than the algorithm of [35] even when restricted to real functions over $\mathbb{Z}_N$, improving on the latter in achieving $\widetilde{\Theta}(\log N)$ rather than $poly(\log N)$ dependency on N.

Our SFT algorithm for functions over product groups $\mathbb{Z}_N^k$ combines our technique developed for functions over $\mathbb{Z}_N$ with techniques similar to those used by [38] for functions over the product group $\mathbb{Z}_2^k$.

Remarks.

1. The SFT algorithm is *non adaptive*, namely, the queries to the function f are independent of the progress of the algorithm.

2. The SFT algorithm is *probabilistic*, that is, it returns the correct output with probability at least $\frac{2}{3}$, where probability is taken over the internal coin tosses of the algorithm. The error probability can be made any arbitrarily small δ with a factor of $\ln\frac{1}{\delta}$ increase in the running time and the sample complexity.

   We point out that there is only one randomized step in the SFT algorithm: the step where we choose the entries $q \in G$ on which to query f.

3. The output of the SFT Algorithm is a list L of elements $\alpha \in G$ for which the Fourier coefficient $\left|\widehat{f}(\alpha)\right|$ is large. Computing ε-approximations[4] for these Fourier coefficients is straightforward. Specifically, the values

   $$val_\alpha = \frac{1}{m}\sum_{i=1}^{m} f(x_i)\chi_\alpha(x_i)$$

   for $m = \Theta(\frac{1}{\varepsilon^2}\ln\frac{\delta}{|L|})$ and $x_1, \ldots, x_m$ chosen uniformly at random from G satisfy with probability at least $1-\delta$ that $\left|val_\alpha - \widehat{f}(\alpha)\right| < \varepsilon$ for all $\alpha \in L$.

[4]We say that val_α is an *ε-approximation* for the Fourier coefficients $\widehat{f}(\alpha)$ if $\left|val_\alpha - \widehat{f}(\alpha)\right| < \varepsilon$.

SFT Applications in Computational Learning Theory

The SFT algorithm has many applications (examples are given in section 1.4, chapter 1). We elaborate here on SFT applications in computational learning.

A common task in computational learning is to find a hypothesis h approximating a function f, when given only samples of the function f. Samples may be given in a variety of forms, *e.g.*, via query access to f. We consider the following variant of this learning problem: f and h are Boolean functions over a finite abelian group $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$. The goal is to find h such that $\|f - h\|_2^2 \leq \gamma$ for $\gamma > 0$ an approximation parameter, and samples of f are given via query access.

A straightforward application of our SFT algorithm gives an efficient solution to the above learning problem, provided that there is a small set $\Gamma \subseteq G$ s.t. $\sum_{\alpha\in\Gamma} \widehat{f}(\alpha)^2 > (1 - \frac{\gamma}{3})$. The learning algorithm operates as follows. It first runs the SFT algorithm to find all $\alpha = (\alpha_1, \ldots, \alpha_k) \in G$ that are $\frac{\gamma}{|\Gamma|}$-significant Fourier coefficients of f along with their $\frac{\gamma}{|\Gamma|}$-approximations val_α, and then returns the hypothesis

$$h(x) \overset{def}{=} \sum_{\alpha \text{ is } \gamma/|\Gamma|\text{-significant}} val_\alpha \cdot \chi_\alpha(x)$$

This hypothesis h satisfies that $\|f - h\|_2^2 \leq \gamma$. The running time of this learning algorithm and the number of queries to f it makes is polynomially bounded in $\log|G|$, $1/\gamma$ and $|\Gamma|$.

We remark that knowing the size of $|\Gamma|$ in inessential for the above sketched algorithm. When $|\Gamma|$ is not known we can apply the above algorithm with doubling guessed values for it, until a value large enough is reached. We can identify that this is the case by estimating $\|f - h\|_2^2$ and verifying that it is indeed less than γ.

More examples of function classes that can be efficiently learned using our SFT algorithm are given in [8].

Organization The rest of this chapter is organized as follows. In section 3.3 we analyze the correctness and running time of the SFT algorithm. Parts of the analysis rely on properties of filter functions $h^{a,b}, h^{\alpha^t,a,b}$ defined there; the properties of these functions are proved in section 3.4.

3.2 The SFT Algorithm

In this section we present the SFT algorithm. We begin with describing the SFT algorithm for functions over $\mathbb{Z}_N$ (in section 3.2.1). We then describe the SFT algorithm for functions over $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ (in section 3.2.2). Finally we conclude with describing the SFT algorithm for functions over arbitrary finite abelian groups (in section 3.2.3).

3.2.1 The SFT Algorithm over $\mathbb{Z}_N$

We first describe the SFT algorithm for functions over $\mathbb{Z}_N$. The input in this case is a group size N, a threshold τ and query access to a function $f\colon \mathbb{Z}_N \to \mathbb{C}$. The output is a short list containing all τ-significant Fourier coefficients, that is, all $\alpha \in \mathbb{Z}_N$ s.t. $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$.

Algorithm overview. The SFT algorithm operates in a binary search fashion. The τ-significant Fourier coefficients may appear anywhere among the elements $\{0,\dots,N-1\}$ of $\mathbb{Z}_N$, and the SFT algorithm "zooms into them" via $\log N$ steps of gradually refining the initial interval $\{0,\dots,N-1\}$ into smaller and small subintervals. At each such refinement step, the SFT algorithm maintains a list of intervals such that (i) each τ-significant Fourier coefficients belongs to some interval in the list, (ii) the list is short, and (iii) the length of the intervals in the list is half the length of the intervals in the list of the previous refinement step. For example, at the first refinement step, the SFT algorithm partitions the initial interval $\{0,\dots,N-1\}$ into two halves $\left\{0,\dots,\frac{N}{2}-1\right\}$ and $\left\{\frac{N}{2},\dots,N-1\right\}$, and decides whether to keep or discard each half by applying a "distinguishing procedure" on it (details below). Likewise, at each subsequent refinement step, the SFT algorithm partitions into two halves each interval in the list maintained by the previous refinements step, and decides whether to keep or discard each half by applying the distinguishing procedure. After $\log_2 N$ refinement steps, the algorithm holds a short list of length 1 intervals that contains all τ-significant coefficients of f. This list is the output of the SFT algorithm.

The distinguishing procedure is the heart of the SFT algorithm. Ideally we'd like the distinguishing procedure to keep an interval iff it contains a τ-significant Fourier coefficient. Such a distinguishing procedure would immediately guarantee property (i) above; moreover, it would also guarantee property (ii) above, because the number of τ-significant Fourier coefficients is not too large.[5] Unfortunately, it is not known how to efficiently compute such a distinguishing procedure. Nevertheless, we present a distinguishing procedure with similar guarantee: it keeps all intervals that contain a τ-significant Fourier coefficient, yet keeping only few intervals.

The distinguishing procedure we present, given an interval $\{a,\dots,b\}$, computes (an approximation of) a weighted sum of squared Fourier coefficients

$$\mathsf{est} \approx \sum_{\alpha\in\mathbb{Z}_N} c_\alpha \cdot \left|\widehat{f}(\alpha)\right|^2$$

such that the weights c_α are high (*i.e.*, close to 1) for α in the interval, and the weights c_α are fast decreasing as α gets farther and farther away from the interval. The distinguishing procedure keeps the interval iff this (approximate) weighted sum is sufficiently large. The threshold for being "sufficiently large" is determined such

[5]Specifically, the number of τ-significant Fourier coefficients is at most $\|f\|_2^2/\tau$. This is because by Parseval identity $\sum_{\alpha\in G}\left|\widehat{f}(\alpha)\right|^2 = \|f\|_2^2$ implying that the number for elements τ-significant α is at most $\tau/\|f\|_2^2$.

that property (i) holds, that is, intervals containing a τ-significant Fourier coefficient are kept. Moreover, with further analysis it is possible to show that property (ii) holds as well, that is, the number of kept intervals is not too large (specifically, it is polynomial in $\|f\|_2^2/\tau$).

A central part of the distinguishing procedure is computing (an approximation of) the weighted sum discussed above. To achieve this, we define a "filter function" h whose (squared) Fourier coefficients are equal to the above coefficients c_α. This allows us to express the above weighted sum as the norm of the convolution of h and f, that is,

$$\sum_{\alpha \in \mathbb{Z}_N} c_\alpha \cdot \left|\widehat{f}(\alpha)\right|^2 = \|h * f\|_2^2$$

Observing that $\|h * f\|_2^2 = \mathbb{E}_{x \in \mathbb{Z}_N} \left(\mathbb{E}_{y \in \mathbb{Z}_N} h(y) f(x-y)\right)^2$, we approximate this value by taking an average over randomly chosen values of x, y in $\mathbb{Z}_N$.

Pseudo-code for the SFT algorithm over $\mathbb{Z}_N$. In algorithms 3.4-3.7 below, we present the SFT algorithm over $\mathbb{Z}_N$ in pseudo-code. Let us briefly describe each of these algorithms.

The SFT Algorithm 3.4 is the main procedure. The input to this algorithm is the value N describing the group $\mathbb{Z}_N$, the threshold τ on the weight of the Fourier coefficients that we seek, a confidence parameter δ s.t. the algorithm succeeds with probability $1-\delta$, and query access to the function $f\colon \mathbb{Z}_N \to \mathbb{C}$ whose Fourier coefficients the algorithm finds. The output of this algorithm is a short list L that contains all τ-significant Fourier coefficients of f with probability at least $1-\delta$. The steps of this algorithm are as follows. The algorithm first calls the Generate Queries Algorithm 3.5 where the set $Q \subseteq \mathbb{Z}_N$ of queries to f is chosen. It then asks and received the value $f(q)$ for each $q \in Q$. Next, the algorithm calls the Fixed Queries SFT Algorithm 3.6 where the search for the significant Fourier coefficients is executed when using only the values $\{(q, f(q))\}_{q \in Q}$. Finally it outputs the list L returned by the Fixed Queries SFT Algorithm 3.6.

In the Generate Queries Algorithm 3.5 the set $Q \subseteq \mathbb{Z}_N$ of queries to f is chosen. The set Q is a random set chosen as follows. The Generate Queries Algorithm 3.5 fixes parameters m_A, m_B, and then chooses a set A of m_A elements uniformly at random from $\mathbb{Z}_N$ and sets $B_1, \ldots, B_{\log N}$ of m_B elements s.t. for each $\ell = 1, \ldots, \log N$, B_ℓ is chosen uniformly at random from $\left\{0, \ldots, 2^{\ell-1}-1\right\}$. The set of queries Q is defined to be

$$Q = A - \bigcup_{\ell=1}^{\log N} B_\ell$$

(where for any two sets X, Y, we denote $X - Y = \{x - y \mid x \in X, y \in Y\}$).

The Fixed Queries SFT Algorithm 3.6 is where the search for the significant Fourier coefficients is executed, when using only the values $\{(q, f(q))\}_{q \in Q}$. The search operates by gradually refining a (short) list $Candidates_\ell$, $\ell = 0, \ldots, \log N - 1$, of intervals $\{a, b\}$ as discussed in the overview above. The list $Candidates_{\log N}$ contains intervals of length 1 corresponding to a list of elements in $\mathbb{Z}_N$ containing all significant

Fourier coefficients of f, which is the output of the algorithm.

The Distinguishing Algorithm 3.7 is the heart of the refining steps in the Fixed Queries SFT Algorithm 3.6. This algorithm, given an interval $\{a, b\}$ of length $N/2^\ell$ and sets A and $B = B_{\ell+1}$ (as were sampled in the Generate Queries Algorithm 3.5), decides whether to keep or discard it. To do this, it computes

$$\mathsf{est}^{a,b} = \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B}\chi_{-\frac{a+b}{2}}(y)f(x-y)\right)^2$$

and decides to keep the interval $\{a, b\}$ iff $\mathsf{est}^{a,b}$ is greater than some threshold. In the analysis we show that (with high probability over the choice of $A, B_{\ell+1}$), $\mathsf{est}^{a,b}$ approximates $\|h * f\|_2^2$ for a filter function h as discussed in the overview, implying that $\{a, b\}$ is kept if it contains a τ-significant Fourier coefficient, and that not too many $\{a, b\}$ are kept.

Algorithm 3.4. SFT
Input: *$N \in \mathbb{N}$, $\tau \in \mathbb{R}^+$, $\delta' \in (0, 1)$ and query access to $f : \mathbb{Z}_N \to \mathbb{C}$.*
Output: *$L \subseteq \mathbb{Z}_N$*
Steps:

1. *Run Generate Queries Algorithm 3.5 on input N, $\frac{\tau}{36}$, $\|f\|_\infty$ and δ for $\delta = \delta'/O\left(\left(\frac{\|f\|_2^2}{\tau}\right)^{1.5}\log N\right)$. Denote its output by $A, B_1, \ldots, B_{\log N}$ and denote $Q = A - \bigcup_\ell^{\log N} B_\ell$*

2. *Query f to find values $f(q)$ for all $q \in Q$*

3. *Run Fixed Queries SFT Algorithm 3.6 on input N, τ, $A, B_1, \ldots, B_{\log N}$ and $\{(q, f(q))\}_{q\in Q}$; denote its output by L.*

4. *Output L*

Algorithm 3.5. Generate Queries Algorithm.
Input: *$N \in \mathbb{N}$, $\gamma \in \mathbb{R}^+$, $\|f\|_\infty$ and $\delta \in (0, 1)$*
Output: *$A, B_1, \ldots, B_{\log N} \subseteq \mathbb{Z}_N$*
Steps:

1. *Let $m_A = \Theta\left(\frac{1}{\gamma^2}\ln\frac{1}{\delta}\right)$ and $m_B = \Theta\left(\frac{1}{\gamma^2}\ln\frac{1}{\delta\gamma}\right)$*

2. *Choose $A \subseteq \mathbb{Z}_N$ a random subset of m_A elements*

3. *For each $\ell = 1, \ldots, \log N$, choose $B_\ell \subseteq \left\{0, \ldots, 2^{\ell-1} - 1\right\}$ be a random subset of $\min\left\{m_B, 2^{\ell-1}\right\}$ elements*

4. *Output $A, B_1, \ldots, B_{\log N}$*

Algorithm 3.6. Fixed Queries SFT Algorithm
Input: $N \in \mathbb{N}$, $\tau \in \mathbb{R}^+$, $A, B_1, \ldots, B_{\log N} \subseteq \mathbb{Z}_N$ *and* $\{(q, f(q))\}_{q \in Q}$ *for* $Q = A - \bigcup_{\ell=1}^{\log N} B_\ell$
Output: $L \subseteq \mathbb{Z}_N$
Steps:

```
1. $Candidate_0 \leftarrow \{\{0, N\}\}$, $\forall \ell = 1, \ldots, \log N$, $Candidate_\ell = \phi$
2. For $\ell = 0, \ldots, \log_2 N - 1$
    (a) For each $\{a', b'\} \in Candidate_\ell$
        For each $\{a, b\} \in \left\{\{a', \frac{a'+b'}{2}\}, \{\frac{a'+b'}{2} + 1, b'\}\right\}$
        i. Run Distinguishing Algorithm 3.7 on input $\{a, b\}$, $\tau$, $A, B_{\ell+1}$, and $\{(q, f(q))\}_{q \in Q}$; denote its output by "decision"
        ii. If decision $= 1$, $Candidate_{\ell+1} \leftarrow Candidate_{\ell+1} \bigcup \{\{a, b\}\}$
3. Output $L = \{\alpha \mid \{\alpha, \alpha\} \in Candidate_{\log N}\}$
```

Algorithm 3.7. Distinguishing Algorithm.
Input: $\{a, b\} \in \mathbb{Z}_N \times \mathbb{Z}_N$, $\tau \in \mathbb{R}^+$, $A, B \subseteq \mathbb{Z}_N$, $\{(q, f(q))\}_{q \in A-B}$
Output: 1 *or* 0
Steps:

```
1. Compute $\mathsf{est}^{a,b} \leftarrow \frac{1}{|A|} \sum_{x \in A} \left( \frac{1}{|B|} \sum_{y \in B} \chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y) f(x - y) \right)^2$
2. If $\mathsf{est}^{a,b} \geq \frac{5}{36}\tau$, decision $= 1$, else decision $= 0$
```

Remark 3.8. *To simplify the presentation of Algorithm 3.5 above, we set the values* m_A, m_B *under the assumption that the function* f *is bounded by some constant, say,* $\|f\|_\infty = 1$. *In general, for unbounded* f, *we set* m_A, m_B *to be* $m_A = \Theta\left(\left(\frac{\|f\|_\infty}{\eta}\right)^2 \ln \frac{1}{\delta}\right)$ *and* $m_B = \Theta\left(\left(\frac{\|f\|_\infty}{\eta}\right)^2 \ln \frac{\|f\|_\infty}{\delta\gamma}\right)$ *for* $\eta = \Theta\left(\min\left\{\gamma, \sqrt{\gamma}, \frac{\gamma}{\|f\|_\infty}\right\}\right)$.

To simplify the presentation of Algorithm 3.6, we ignored the question of whether $\frac{a'+b'}{2}$ *is an integer. More precisely, instead of taking* $\frac{a'+b'}{2}$ *as written above, we partition* $\{a', \ldots, b'\}$ *into two disjoint subintervals* $\{a', \ldots, c\}, \{c+1, \ldots, b'\}$ *of roughly the same length, namely* $-1 \leq |\{c+1, \ldots, b'\}| - |\{a', \ldots, c\}| \leq 1$, *and proceed with* $\mathsf{est}^{a',c}$ *and* $\mathsf{est}^{c+1,b'}$. *Moreover, to ease the analysis we make sure that at each iteration* ℓ, *all intervals are of length exactly* $\lceil (N/2^\ell) \rceil$. *To achieve this we reorganize the intervals (possibly adding a few extra elements).*

3.2.2 The SFT Algorithm over $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$

We next describe the SFT algorithm for functions over $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$. The input in this case is a description of the group by $N_1, \ldots, N_k$, a threshold τ and query access to a function $f\colon G \to \mathbb{C}$. The output is a short list containing all τ-significant Fourier coefficients, that is, all $\alpha \in G$ s.t. $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$.

Algorithm overview. The SFT algorithm finds the τ-significant Fourier coefficients $(\alpha_1, \ldots, \alpha_k) \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ by gradually revealing its coordinates one after the other. At the first step, the algorithm finds the first coordinates of all the τ-significant Fourier coefficients, that is, it finds length 1 prefixes of the τ-significant Fourier coefficients. At the second step, the algorithm extends each length 1 prefix to all its continuation into length 2 prefixes of the τ-significant Fourier coefficients. The algorithm continues in extending prefixes of the τ-significant Fourier coefficients one coordinate at a time. After k step, the algorithm holds length k prefixes, which are the list of τ-significant Fourier coefficients.

To extend a length $t-1$ prefix $(\alpha_1, \ldots, \alpha_{t-1})$ of a τ-significant Fourier coefficient to a prefix of length t, the algorithm searches for all values α_t of the t-th coordinate such that $(\alpha_1, \ldots, \alpha_{t-1}, \alpha_t)$ is a length t prefixes of a τ-significant Fourier coefficient. This search is done in a binary search fashion, similarly to the SFT algorithm for functions over $\mathbb{Z}_{N_t}$. Namely, the search proceeds by gradually refining the initial interval $\{0, \ldots, N_t\}$ into smaller and smaller subintervals, each time applying a distinguishing procedure to decide whether to keep or discard a subinterval.

The distinguishing procedure we use here is different than the one used for the case of functions over $\mathbb{Z}_N$. Ideally we'd like the distinguishing procedure to keep an interval iff it contains α_t such that $(\alpha_1, \ldots, \alpha_{t-1}, \alpha_t)$ is a length t prefix of a a τ-significant Fourier coefficient. It is not known how to efficiently compute such a distinguishing procedure. Nevertheless, we present a distinguishing procedure with similar guarantee: it keeps all intervals that contain a τ-significant Fourier coefficient, yet keeping only few intervals. Specifically, the distinguishing procedure, given a length $t-1$ prefix $\alpha = (\alpha_1, \ldots, \alpha_{t-1})$ and an interval $\{a, \ldots, b\}$, computes (an approximation of) a weighted sum of squared Fourier coefficients

$$\mathsf{est} \approx \sum_{\alpha_t \in \mathbb{Z}_{N_t}} c_{\alpha_t} \cdot \sum_{\alpha' \in \mathbb{Z}_{N_{t+1}} \times \ldots \times \mathbb{Z}_{N_k}} \left| \widehat{f}(\alpha \alpha_t \alpha') \right|^2$$

such that the weights c_{α_t} are high (*i.e.*, close to 1) for α_t in the interval, and the weights c_{α_t} are fast decreasing as α gets farther and farther away from the interval. The distinguishing procedure keeps the interval iff this (approximate) weighted sum is sufficiently large. To compute (an approximation of) this weighted sum, we define a "filter function" h whose (squared) Fourier coefficients $\left|\widehat{h}(\beta)\right|^2$ are equal to the above coefficients c_{α_t} when the length t prefix of β is the given prefix α, and they are zero otherwise. With this filter function we express the above weighted sum as the norm of the convolution of h and f, which we in turn approximate by taking an average over randomly chosen values

Pseudo-code for SFT Algorithm over $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$. In Algorithms 3.9-3.12 below, we present the SFT algorithm for functions over $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ in pseudo-code.

The overall structure of the algorithm is similar to the one for functions over $\mathbb{Z}_N$. There are differences in the Fixed Queries SFT Algorithm 3.11 and in the Distinguishing Algorithm 3.7. In the Fixed Queries SFT Algorithm 3.11 there is an additional outer loop iterating over $t = 1, \ldots, k$ in which we gradually construct lists length t prefixes of τ-significant Fourier coefficients of f. In the Distinguishing Algorithm 3.7, given a length $t-1$ prefix $\alpha^t = (\alpha_1, \ldots, \alpha_{t-1})$ and interval $\{a, \ldots, b\} \subseteq \mathbb{Z}_{N_t}$, the task is to distinguish whether or not there is a τ-significant Fourier coefficient β whose length t prefix is $\alpha^t \alpha_t$ for some $\alpha_t \in \{a, \ldots, b\}$. This is achieved as discussed in the overview above.

Algorithm 3.9. SFT
Input: *A description* $\{(1, N_i)\}_{i=1}^k$ *of the group* $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, $\tau \in \mathbb{R}^+$, $\delta' \in (0,1)$, *and query access to* $f \colon G \to \mathbb{C}$.
Output: $L \subseteq \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$
Steps:

1. *Run Generate Queries Algorithm 3.10 on input* $\{(1, N_i)\}_{i=1}^k$, $\frac{\tau}{36}$, $\|f\|_\infty$ *and* $\delta'/\Theta\left(\frac{1}{\tau}\left(\frac{\|f\|_2^2}{\tau}\right)^{1.5} \log|G|\right)$. *Denote its output by* A, $\{B_{t,\ell}\}_{t\in[k],\ell\in[\log N_t]}$, *and denote* $Q = A - \bigcup_{t\in[k],\ell\in[\log N_t]} B_{t,\ell}$
2. *Query* f *to find values* $f(q)$ *for all* $q \in Q$
3. *Run Fixed Queries SFT Algorithm 3.11 on input* $\{(1, N_i)\}_{i=1}^k$, τ, A, $\{B_{t,\ell}\}_{t\in[k],\ell\in[\log N_t]}$ *and* $\{(q, f(q))\}_{q\in Q}$ *for* $Q = A - \bigcup_{t\in[k],\ell\in[\log N_t]} B_{t,\ell}$. *Denote its output by* L.
4. *Output* L

Algorithm 3.10. Generate Queries Algorithm.
Input: *A description* $\{(1, N_i)\}_{i=1}^k$ *of the group* $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, $\gamma \in \mathbb{R}^+$, $\|f\|_\infty$ *and* $\delta \in (0,1)$
Output: $A, \{B_{t,\ell}\}_{t\in[k],\ell\in[\log N_t]}$
Steps:

1. *Let* $m_A = \Theta\left(\frac{1}{\gamma^2} \ln \frac{1}{\delta}\right)$ *and* $m_B = \Theta\left(\frac{1}{\gamma^2} \ln \frac{1}{\delta\gamma}\right)$
2. *Choose* $A \subseteq G$ *a random subset of* m_A *elements*
3. *For each* $t \in [k]$, $\ell \in [\log N_t]$, *choose* $B \subseteq \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}} \times \{0, \ldots, 2^{\ell-1} - 1\} \times \{0\} \times \ldots \times \{0\}$ *a random subset of* m_B *elements*
4. *Output* A, $\{B_{t,\ell}\}_{t\in[k],\ell\in[\log N_t]}$

Algorithm 3.11. Fixed Queries SFT Algorithm
Input: *A description* $\{(1, N_i)\}_{i=1}^k$ *of the group* $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, $\tau \in \mathbb{R}^+$, A, $\{B_{t,\ell}\}_{t\in[k],\ell\in[\log N_t]}$ *and* $\{(q, f(q))\}_{q\in Q}$ *for* $Q = A - \bigcup_{t\in[k],\ell\in[\log N_t]} B_{t,\ell}$
Output: $L \subseteq \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$
Steps:

1. *Let* $Prefixes_0 = \{$*the empty string*$\}$, $Prefixes_1, \ldots, Prefixes_k = \phi$
2. *For* $t = 1, \ldots, k$

```
(a) For each $\alpha^t = (\alpha_1, \ldots, \alpha_{t-1}) \in Prefixes_{t-1}$
    i. $Candidate_{\alpha^t,0} \leftarrow \{\{0, N_t\}\}$, $\forall \ell = 1, \ldots, \log N_\ell$, $Candidate_{\alpha^t,\ell} = \phi$
    ii. For $\ell = 0, \ldots, \log_2 N_t - 1$
        A. For each $\{a', b'\} \in Candidate_{\alpha^t,\ell}$
           For each $\{a, b\} \in \left\{\left\{a', \frac{a'+b'}{2}\right\}, \left\{\frac{a'+b'}{2} + 1, b'\right\}\right\}$
           • Run the Distinguishing Algorithm 3.12 on input $\alpha^t$, $\{a, b\}$, $\tau$,
             $A, B_{t,\ell+1}$ and $\{(q, f(q))\}_{q \in A \times B_{t,\ell+1}}$; denote its outputs be "decision"
           • If decision = 1, $Candidate_{\alpha^t,\ell+1} \leftarrow Candidate_{\alpha^t,\ell+1} \bigcup \{\{a, b\}\}$
    iii. For each $\{a, a\} \in Candidate_{\alpha^t,\log N_t}$ denote $\alpha^t a = (\alpha_1, \ldots, \alpha_{t-1}, a)$.
         Let
             $L_t(\alpha^t) = \left\{ \alpha^t a \;\middle|\; \{a, a\} \in Candidates_{\alpha^t,\log N_t} \right\}$
(b) Let $Prefixes_t \leftarrow \bigcup_{\alpha^t \in Prefixes_{t-1}} L(\alpha^t)$
3. Output $Prefixes_k$
```

Algorithm 3.12. Distinguishing Algorithm.
Input: $\alpha^t \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$, $\{a, b\} \in \mathbb{Z}_{N_t} \times \mathbb{Z}_{N_t}$, $\tau \in \mathbb{R}^+$, $A, B \subseteq G$ and $\{(q, f(q))\}_{q \in A - B}$.
Output: 1 *or* 0
Steps:

1. *Compute*

$$\mathsf{est}^{\alpha^t,a,b} \leftarrow \frac{1}{|A|} \sum_{x \in A} \left(\frac{1}{|B|} \sum_{y \in B} \chi_{\alpha^t}(y^t) \chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y_t) \cdot f(x - y) \right)^2$$

   for $\chi_{\alpha^t}(y^t) = \prod_{j=1}^{t-1} e^{i\frac{2\pi}{N_j} \cdot \alpha^t_j y^t_j}$ *an evaluation of the* α^t *character of the group* $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$*, and* $\chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y_t) = e^{-i\frac{2\pi}{N_t} \cdot \lfloor(\frac{a+b}{2})\rfloor y_t}$ *an evaluation of the* $-\lfloor(\frac{a+b}{2})\rfloor$ *character of the group* $\mathbb{Z}_{N_t}$.

2. *If* $\mathsf{est}^{\alpha^t,a,b} \geq \frac{5}{36}\tau$*, decision* = 1*, else decision* = 0

Remark 3.13. *To simplify the presentation of Algorithm 3.10 above, we set the values* m_A, m_B *under the assumption that the function* f *is bounded by some constant, say,* $\|f\|_\infty = 1$*. In general, for unbounded* f*, we set* m_A, m_B *to be* $m_A = \Theta\left(\left(\frac{\|f\|_\infty}{\eta}\right)^2 \ln \frac{1}{\delta}\right)$ *and* $m_B = \Theta\left(\left(\frac{\|f\|_\infty}{\eta}\right)^2 \ln \frac{\|f\|_\infty}{\delta\gamma}\right)$ *for* $\eta = \Theta\left(\min\left\{\gamma, \sqrt{\gamma}, \frac{\gamma}{\|f\|_\infty}\right\}\right)$.

To simplify the presentation of Algorithm 3.11, we ignored the question of whether $\frac{a'+b'}{2}$ *is an integer. More precisely, instead of taking* $\frac{a'+b'}{2}$ *as written above, we partition* $\{a', \ldots, b'\}$ *into two disjoint subintervals* $\{a', \ldots, c\}, \{c+1, \ldots, b'\}$ *of roughly the same length, namely* $-1 \leq |\{c+1, \ldots, b'\}| - |\{a', \ldots, c\}| \leq 1$*, and proceed with* $\mathsf{est}^{a',c}$ *and* $\mathsf{est}^{c+1,b'}$*. Moreover, to ease the analysis we make sure that at each iteration* ℓ*, all intervals are of length exactly* $\lceil (N/2^\ell) \rceil$*. To achieve this we reorganize the intervals (possibly adding a few extra elements).*

3.2.3 The SFT Algorithm over Finite Abelian Groups

The SFT Algorithm for arbitrary finite abelian groups G is defined by utilizing the isomorphism between G and a direct product group.[6] as follows.

Given a description $\{(g_j, N_j)\}_{j=1}^k$ of the group G, a threshold τ and query access to a function $f\colon G \to \mathbb{C}$, we simulate query access to a function f' over a direct product group isomorphic to G, and apply the SFT Algorithm 3.9 on input a description of the direct product group, the threshold τ and query access to f'. Output $L = \left\{ \prod_{j=1}^k g_j^{x_j} \,\middle|\, (x_1, \ldots, x_k) \in L' \right\}$ for L' the output of the SFT Algorithm 3.9.

To complete the description of the algorithm we define the function f' and explain how to efficiently simulate query access to f' when given query access to f. The function f' is defined by $f'(x_1, \ldots, x_k) = f(\prod_{j=1}^k g_j^{x_j})$. The function f' is computable in time polynomial in $\log |G|$.

3.3 Analysis of the SFT Algorithm

In this section we analyze the SFT algorithm. Analyze the SFT algorithm for functions over $\mathbb{Z}_N$ in section 3.3.1; analyzing the SFT algorithm for functions over $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ in section 3.3.2; and analyzing the SFT algorithm for functions over arbitrary finite abelian groups in section 3.3.3.

3.3.1 Analysis of the SFT Algorithm over $\mathbb{Z}_N$

In this section we analyze the SFT Algorithm 3.4, that is, the SFT algorithm for complex functions f over $\mathbb{Z}_N$.

To prove the correctness of the SFT algorithm, we first define a filter function $h^{a,b}$, and define a criterion on when the sets $A, B_1, \ldots, B_{\log N}$ chosen by the Generate Queries Algorithm 3.5 are "good". We then show in Theorem 3.16 that the SFT algorithm succeeds if the set $A, B_1, \ldots, B_{\log N}$ chosen by the Generate Queries Algorithm 3.5 are good, and show moreover, that the event of $A, B_1, \ldots, B_{\log N}$ being good happens with high probability (where the probability is taken over the random coins of the Generate Queries Algorithm 3.5).

The filter function $h_N^{a,b}$ that we define is a phase shift of a periodic square function defined as follows. For $\{-a, \ldots, a\}$ a symmetric interval around zero,[7] $h_N^{-a,a}(y) = 2a$ is constant for all $y \in \{0, \ldots, N/2a\}$, and $h_N^{-a,a}(y) = 0$ otherwise. For general intervals $\{a, \ldots, b\}$, $h_N^{a,b}(y) = \chi_{-s}(y) h_N^{-s,s}(y)$ is a phase shift of $h_N^{-s,s}$ by a shift $s = \frac{b+a}{2}$ chosen to translate the center of the interval $\{a, \ldots, b\}$ to zero.

[6]Recall that if G a finite abelian group generated by $g_1, \ldots, g_k$ of orders $N_1, \ldots, N_k$, respectively, then G is isomorphic to the direct product group $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ by mapping $(x_1, \ldots, x_k) \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ to $\prod_{j=1}^k g_j^{x_j} \in G$.

[7]For any $a \in \mathbb{Z}_N$, we denote by $-a$ the group element $N - a$.

Definition 3.14 ($h_N^{a,b}$)**.** *For any positive integer N and $a \leq b$ in $\mathbb{Z}_N$, we define a filter function $h_N^{a,b} \colon \mathbb{Z}_N \to \mathbb{C}$ as follows. If $a < b$,*

$$h_N^{a,b}(y) \overset{def}{=} \begin{cases} \frac{N}{t} \cdot \chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y) & \text{if } y \in \{0, \dots, t-1\} \\ 0 & \text{otherwise} \end{cases}$$

for $t = \lfloor(\frac{N}{2(b-a)})\rfloor$ and $\chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y) = e^{i\frac{2\pi}{N}\lfloor(\frac{a+b}{2})\rfloor y}$ a character in the group $\mathbb{Z}_N$. If $a = b$,

$$h_N^{a,a}(y) = \chi_{-a}(y)$$

When N is clear from the context we often omit it and denote $h^{a,b} = h_N^{a,b}$.

We say that the sets $A, B_1, \dots, B_{\log N}$ chosen by the Generate Queries Algorithm 3.5 are *good*, if the values $\mathsf{est}^{a,b}$ computed by the Distinguishing Algorithm 3.7 throughout the execution of the SFT algorithm all satisfy that

$$\mathsf{est}^{a,b} \approx \|h^{a,b} * f\|_2^2$$

for f the input function.

Definition 3.15 (Good A, B)**.** *For every positive integer N, a function $f \colon \mathbb{Z}_N \to \mathbb{C}$, subsets $A, B \subseteq \mathbb{Z}_N$, an interval $\{a, \dots, b\}$ in $\mathbb{Z}_N$, and a threshold $\gamma \in \mathbb{R}^+$, we say that A, B are γ-*good w.r. to f, a, b, *if*

$$\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right| < \gamma$$

for $\mathsf{est}^{a,b} = \frac{1}{|A|}\sum_{x \in A}\left(\frac{1}{|B|}\sum_{y \in B}\chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y) f(x-y)\right)^2$ the value computed by the Distinguishing Algorithm 3.7 on input a, b, τ, A, B and $\{(q, f(q))\}_{q \in A-B}$.

We say that a call to the Distinguishing Algorithm with parameters $\{a, b\}$, τ, A, B, $\{(q, f(q))\}_{q \in A-B}$ is good *if A, B are $\frac{\tau}{36}$-good w.r. to f, a, b.*

Analyzing SFT Algorithm 3.4

In Theorem 3.16 below we show the SFT algorithm returns a short list containing all significant Fourier coefficients of f, and that its running time is polynomial in $\log N, \frac{1}{\tau}$.

The proof of Theorem 3.16 builds on three lemmata: In Lemma 3.20 we show that the sets $A, B_1, \dots, B_{\log N}$ chosen by the Generate Queries Algorithm 3.5 are good with high probability. In Lemma 3.18 we show that when $A, B_1, \dots, B_{\log N}$ are good then all calls to the Distinguishing Algorithm 3.7 are good. In Lemma 3.17 we show that when all calls to the Distinguishing Algorithm 3.7 are good, then the Fixed Queries SFT Algorithm 3.6 outputs a short list containing every significant Fourier coefficient of f. This in turn implies that the SFT Algorithm 3.4 outputs a short list containing every significant Fourier coefficient of f.

Theorem 3.16 (Analyzing SFT Algorithm 3.4). *Let the input to the SFT Algorithm 3.4 be $N \in \mathbb{N}, \tau \in \mathbb{R}^+, \delta' \in (0,1)$ and query access to $f\colon \mathbb{Z}_N \to \mathbb{C}$; denote its output by L. Then, with probability at least $1-\delta'$, the following hold:*

1. *$L \supseteq \mathsf{Heavy}_\tau(f)$, i.e., L contains all τ-significant Fourier coefficients of f*
2. *$|L| \leq O(1/\tau)$, i.e., L is short*
3. *The running time of the SFT Algorithm 3.4 is at most polynomial in $\log N, \frac{1}{\tau}$*

Remark. Details on the running time of the SFT Algorithm 3.4 are as follows. For Boolean f the running time is at most $\widetilde{\Theta}\left(\frac{\log N}{\tau^{5.5}} \cdot \ln \frac{1}{\delta'}\right)$. For general functions f, the running time is at most $\widetilde{\Theta}\left(\log N \cdot \left(\frac{\|f\|_2^2}{\tau}\right)^{1.5} \cdot \left(\frac{\|f\|_\infty^2}{\eta^2} \cdot \ln \frac{1}{\delta}\right)^2\right)$ for $\eta = \Theta\left(\min\left\{\tau, \sqrt{\tau}, \frac{\tau}{\|f\|_\infty}\right\}\right)$ and $\delta = \delta'/O\left(\left(\frac{\|f\|_2^2}{\tau}\right)^{1.5} \log N\right)$.

Proof. Let δ be as set in step 1 of the SFT Algorithm 3.4. Below we show that all calls to the Distinguishing Algorithm in step 2(a)i of the Fixed Queries Algorithm are good with probability at least $1 - \log N \cdot \Theta\left((\|f\|_2^2/\tau)^{1.5}\right) \cdot \delta$. Assigning $\delta = \delta'/\left(\log N \cdot \Theta\left((\|f\|_2^2/\tau)^{1.5}\right)\right)$ as was set in step 1 of the SFT Algorithm we get that the above all calls to the Distinguishing Algorithm in step 2(a)i of the Fixed Queries Algorithm are good with probability at least $1-\delta'$. By Lemma 3.17 this concludes the proof.

In the following we abbreviate by saying "call" to refer to a call to the Distinguishing Algorithm in step 2(a)i of the Fixed Queries Algorithm. By Lemma 3.20, each single call is good with probability at least $1-\delta$. We show that all calls are good with high probability using induction and union bound as follows. For $\ell = 0$, $|Candidate_0| = 1$, so there is a single call, an by the above it is good with probability at least $1-\delta$. Assuming all calls in iterations $0, \ldots, \ell-1$ were good, we show that all calls in iteration ℓ are good as well. By Lemma 3.17, $|Candidate_\ell| \leq \Theta\left((\|f\|_2^2/\tau)^{1.5}\right)$, so by union bound the calls for all $\{a,b\} \in Candidate_\ell$ are good with probability at least $1 - \Theta\left((\|f\|_2^2/\tau)^{1.5}\right) \cdot \delta$. Finally, applying union bound over all iterations $\ell = 0, \ldots, \log N - 1$, we conclude that all calls are good with probability at least $1 - \log N \cdot \Theta\left((\|f\|_2^2/\tau)^{1.5}\right) \cdot \delta$. □

Analyzing Fixed Queries SFT Algorithm 3.6

In Lemma 3.17 below we show that when all calls to the Distinguishing Algorithm 3.7 are good, then the Fixed Queries SFT Algorithm 3.6 outputs a short list containing every significant Fourier coefficient of f.

We prove this lemma by induction, showing that for every ℓ, the list of intervals $Candidate_\ell$ which is maintained by the Fixed Queries SFT Algorithm 3.6 satisfy that every significant Fourier coefficient belongs to some interval in the list, and moreover, the number of intervals in the list is not too large. The details of the induction are given in Lemma 3.17.1.

Lemma 3.17 (Analyzing Fixed Queries SFT Algorithm 3.6)**.** *Let the input to the Fixed Queries Algorithm 3.6 be $N \in \mathbb{N}$, $\tau \in \mathbb{R}^+$, $A, B_1, \ldots, B_{\log N} \subseteq \mathbb{Z}_N$ and $\{(q, f(q))\}_{q \in Q}$ for $Q = A - \bigcup_{\ell}^{\log N} B_\ell$; denote its output by L. If all calls to the Distinguishing Algorithm in step 2(a)i of the Fixed Queries Algorithm are good, then the following holds:*

1. *$L \supseteq \mathsf{Heavy}_\tau(f)$, i.e., L contains all τ-significant Fourier coefficients of f*
2. *$|L| \leq O(1/\tau)$, i.e., L is short, and*
3. *The running time of the Fixed Queries SFT Algorithm 3.6 is at most $\Theta\left(\log N \cdot \left(\frac{\|f\|_2^2}{\tau}\right)^{1.5} \cdot |Q|\right)$*

Proof. Let $Candidate_\ell$ be as defined in Fixed Queries SFT Algorithm 3.4. Observe that each $Candidate_\ell$ consists of intervals on length $N/2^\ell$. In Lemma 3.17.1 below we show that $\forall \ell = 1, \ldots, \log N$, $Candidate_\ell$ satisfies the following:
(1) $\bigcup_{\{a,b\} \in Candidate_\ell} \{a, \ldots, b\} \supseteq \mathsf{Heavy}_\tau(f)$, and
(2) $|Candidate_\ell| \leq \Theta\left((\|f\|_2^2/\tau)^{1.5}\right)$.
In particular, this implies that $Candidate_{\log N}$ is a list of intervals, each of length 1, s.t. $\{\alpha \mid \{\alpha, \alpha\} \in Candidate_{\log N}\} \supseteq \mathsf{Heavy}_\tau(f)$, and therefore

$$L = \{a \mid \{a, a\} \in Candidate_{\log N}\} \supseteq \mathsf{Heavy}_\tau(f)$$

Moreover, by Lemma 3.17.1, for all $\{a, a\} \in Candidate_{\log N}$, a is $O(\tau)$-significant Fourier coefficient of f. This implies that there are at most $O(1/\tau)$ such intervals in $Candidate_{\log N}$. Therefore, the output list $L = \{a \mid \{a, a\} \in Candidate_{\log N}\}$ is of size at most $O(1/\tau)$.

The running time is $T = \sum_{\ell=1}^{\log N} |Candidate_\ell| \cdot |A| \cdot |B_\ell|$. Assigning the bound on $|Candidate_\ell|$ from Lemma 3.17.1, we get that

$$T \leq \Theta\left(|Q| \cdot \left(\frac{\|f\|_2^2}{\tau}\right)^{1.5} \log N\right)$$

Lemma 3.17.1. *Let $\ell \in 0, \ldots, \log N - 1$. If for every $\ell' < \ell$, in all iterations ℓ' all call to the Distinguishing Algorithm in step 2(a)i of the Fixed Queries Algorithm were good, then the following holds for the ℓ-th iteration:*

1. *$\bigcup_{\{a,b\} \in Candidate_\ell} \{a, \ldots, b\} \supseteq \mathsf{Heavy}_\tau(f)$*
2. *$|Candidate_\ell| \leq \Theta\left((\|f\|_2^2/\tau)^{1.5}\right)$*

Moreover, for $\ell = \log N$, $\forall \{a, a\} \in Candidate_{\log N}$, a is $O(\tau)$-significant Fourier coefficient of f.

Proof. We prove by induction that the two conditions in the above invariant hold. At the first step of the algorithm $Candidate_0 = \{\{0, N-1\}\}$, which trivially satisfies the above two conditions.

Next we show that if the conditions of the theorem hold for $Candidate_\ell$, then they also hold for $Candidate_{\ell+1}$. First we prove that condition 1 holds. If $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$, then by the induction hypothesis there exists a pair $\{a, b\} \in Candidate_\ell$ s.t. $\alpha \in \{a, \ldots, b\}$. Trivially, either $\alpha \in \left\{a, \ldots, \frac{a+b}{2}\right\}$ or $\alpha \in \left\{\frac{a+b}{2}+1, \ldots, b\right\}$. If $\alpha \in \left\{a, \ldots, \frac{a+b}{2}\right\}$, then by Lemma 3.18, *decision* $= 1$ and therefore, $\left\{a, \frac{a+b}{2}\right\} \in Candidate_{\ell+1}$. Similarly, if $\alpha \in \left\{\frac{a+b}{2}+1, \ldots, b\right\}$, then $\left\{\frac{a+b}{2}+1, b\right\} \in Candidate_{\ell+1}$. Namely,

$$\bigcup_{\{a,b\} \in Candidate_\ell} \{a, \ldots, b\} \supseteq \mathsf{Heavy}_\tau(f)$$

Second, we prove that condition 2 holds. Recall that $\{a, b\} \in Candidate_{\ell+1}$ implies that in the ℓ-th iteration of main loop of Algorithm 3.6 *decision* $= 1$. By Lemma 3.18, this implies that $\sum_{\alpha \in Ext^{a,b}} \left|\widehat{f}(\alpha)\right|^2 \geq \tau/12$ (for $Ext^{a,b} = \{\alpha \mid \mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \Delta \cdot 2(b-a)\}$ where $\Delta = \sqrt{\frac{24}{\tau}} \|f\|_2$ as in Theorem 3.18). By Parseval identity, there are at most $\frac{12}{\tau}\|f\|_2^2$ intervals $\{a, \ldots, b\}$ with disjoint extensions $Ext^{a,b}$ such that the weight $\sum_{\alpha \in Ext^{a,b}} \left|\widehat{f}(\alpha)\right|^2$ is at least $\tau/12$. Each such extension $Ext^{a,b}$ intersects with at most $2\Delta + 1$ disjoint intervals $\{a', \ldots, b'\}$ of size $b - a$. Implying that

$$|Candidate_{\ell+1}| \leq \frac{12}{\tau}\|f\|_2^2 \cdot (2\Delta + 1) = \Theta\left(\left(\|f\|_2^2/\tau\right)^{1.5}\right)$$

Finally, we show that $|Candidate_{\log N}| \leq O(1/\tau)$. Recall that for all $\{a, a\} \in Candidate_{\log N}$, $A, B_{\log N}$ are $\tau/36$-good w.r. to f, a, a, *i.e.*, $|\mathsf{est}^{a,a} - \|h^{a,a} * f\|_2^2| < \tau/36$. By Proposition 3.31 $\|h^{a,a} * f\|_2^2 = \left|\widehat{f}(a)\right|^2$. Combining the two we conclude that $\left|\mathsf{est}^{a,a} - \left|\widehat{f}(a)\right|^2\right| < \tau/36$. Namely, for each $\{a, a\} \in Candidate_{\log N}$, a is $\frac{35}{36}\tau$-significant. □
□

Analyzing Distinguishing Algorithm 3.7

In Lemma 3.18 we show that when the sets chosen by the Generate Queries Algorithm 3.5 are good, then the Distinguishing Algorithm keeps all intervals $\{a, b\}$ that contain a significant Fourier coefficient of f, and moreover, it keeps not too many intervals.

The proof of Lemma 3.18 relies on properties of the filter $h^{a,b}$. These properties are stated in Lemma 3.19 below. The proof of these properties is deferred to section 3.4.1.

Lemma 3.18 (Analyzing Distinguishing Algorithm 3.7)**.** *Let the input to the Distinguishing Algorithm 3.7 be $\{a, b\} \in \mathbb{Z}_N \times \mathbb{Z}_N$, $\tau \in \mathbb{R}^+$, $A, B \subseteq \mathbb{Z}_N$ s.t. that A, B*

are $\tau/36$-good w.r. to f, a, b, and $\{(q, f(q))\}_{q\in A-B}$; denote its output by "decision". Then the following holds:

- *If $\exists\alpha \in \{a, \ldots, b\}$ with $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$, then decision $= 1$*
- *If decision $= 1$, then $\sum_{\alpha\in Ext^{a,b}} \left|\widehat{f}(\alpha)\right|^2 \geq \tau/12$ for $Ext^{a,b} = \{\alpha \mid \mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \Delta \cdot 2(b-a)\}$ an extension of the interval $\{a, \ldots, b\}$ defined with respect to a parameter $\Delta = \sqrt{\frac{24}{\tau}}\|f\|_2$.*

Proof. Let $\mathsf{est}^{a,b} = \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B}\chi_{-\frac{a+b}{2}}(y)f(x-y)\right)^2$ as defined in the Distinguishing Algorithm 3.7. Since A, B are $\tau/36$-good w.r. to f, a, b then

$$\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right| < \tau/36$$

Recall that the Distinguishing Algorithm outputs *decision* $= 1$ iff $\mathsf{est}^{a,b} > \frac{5}{36}\tau$. By Lemma 3.19 when assigning $\gamma = \tau/36$ and $\lambda = \frac{5}{6}\tau$, the conditions of this lemma hold, namely, if $\exists\alpha_0 \in \{a, \ldots, b\}$ with $\left|\widehat{f}(\alpha_0)\right|^2 \geq \tau$, then $\mathsf{est}^{a,b} \geq \frac{5}{36}\tau$, and if $\mathsf{est}^{a,b} \geq \frac{5}{36}\tau$, then $\sum_{\alpha\in Ext^{a,b}} \left|\widehat{f}(\alpha)\right|^2 \geq \frac{1}{12}\tau$. □

Lemma 3.19 (Properties of $\|h^{a,b} * f\|_2^2$)**.** *For any $\gamma > 0$, $f\colon \mathbb{Z}_N \to \mathbb{C}$ and $\mathsf{est}^{a,b} \in \mathbb{R}$, if $\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right| \leq \gamma$, then the following holds:*

- *$\mathsf{est}^{a,b} \geq \sum_{\alpha\in\{a,\ldots,b\}} \frac{\left|\widehat{f}(\alpha)\right|^2}{6} - \gamma$. In particular, if $\exists\alpha_0 \in \{a, \ldots, b\}$ with $\left|\widehat{f}(\alpha_0)\right|^2 \geq \tau$, then $\mathsf{est}^{a,b} \geq \frac{\tau}{6} - \gamma$*
- *For all $\lambda > 0$, if $\mathsf{est}^{a,b} \geq \lambda$, then $\sum_{\alpha\in Ext^{a,b}} \left|\widehat{f}(\alpha)\right|^2 \geq \lambda - 2\gamma$ for $Ext^{a,b} = \left\{\alpha \mid \mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \Delta \cdot 2(b-a)\right\}$ an extension of the interval $\{a, \ldots, b\}$ defined with respect to a parameter $\Delta = \sqrt{\frac{2}{3\gamma}}\|f\|_2$.*

Proof. Proof appears in section 3.4.1. □

Analyzing Generate Queries Algorithm 3.5

In Lemma 3.20 we show that the sets $A, B_1, \ldots, B_{\log N}$ chosen by the Generate Queries Algorithm 3.5 are good with high probability.

Lemma 3.20 (Analyzing Generate Queries Algorithm 3.5)**.** *Let the input to the Generate Queries Algorithm 3.5 be $N \in \mathbb{N}$, $\gamma \in \mathbb{R}^+$, $\|f\|_\infty$ and $\delta \in (0,1)$ Denote its output by A, $B_1, \ldots, B_{\log N}$ and $Q = A - \bigcup_{\ell=1}^{\log N} B_\ell$. Then for each $\ell \in \{0, \ldots, \log N - 1\}$ and $a, b \in \mathbb{Z}_N$ s.t. $b - a = \lceil(N/2^\ell)\rceil$, A, B_ℓ are γ-good w.r. to f, a, b with probability at least $1 - \delta$.*

Proof. Fix $\ell \in \{0, \dots, \log N - 1\}$ and a, b s.t. $b - a = \lceil (N/2^\ell) \rceil$. Applying Lemma 3.21 below, with γ, δ and A, B_ℓ as in Generate Queries Algorithm 3.5 we conclude that A, B are γ-good w.r. to f, a, b with probability at least $1 - \delta$. $\square$

Lemma 3.21. *Let $\ell \in \{0, \dots, \log N - 1\}$, $a, b \in \mathbb{Z}_N$ s.t. $b - a = \lceil (N/2^\ell) \rceil$, A a random subset of $\mathbb{Z}_N$ and B a random subset of $\left\{0, \dots, 2^{\ell-1} - 1\right\}$. For any $\gamma > 0$ and $\delta \in (0,1)$, denote $\eta = O\left(\min\left\{\gamma, \sqrt{\gamma}, \frac{\gamma}{\|f\|_\infty}\right\}\right)$, if $|A| = \Theta\left(\left(\frac{\|f\|_\infty}{\eta}\right)^2 \ln \frac{1}{\delta}\right)$ and $|B| = \Theta\left(\left(\frac{\|f\|_\infty}{\eta}\right)^2 \ln \frac{\|f\|_\infty}{\delta\gamma}\right)$ are sufficiently large, then A, B are γ-good w.r. to f, a, b with probability at least $1 - \delta$.*

Remark. *In particular for Boolean f: for any $\gamma > 0$ and $\delta \in (0,1)$, if $|A| = \Theta\left(\frac{1}{\gamma^2} \ln \frac{1}{\delta}\right)$ and $|B| = \Theta\left(\frac{1}{\gamma^2} \ln \frac{1}{\delta\gamma}\right)$, then A, B are γ-good w.r. to f, a, b with probability at least $1 - \delta$.*

Proof. For every $x \in A$, denote $\mathsf{est}^{a,b}(x) = \frac{1}{|B|} \sum_{y \in B} \chi_{-\lfloor (\frac{a+b}{2}) \rfloor}(y) f(x - y)$. Let $\mathsf{est}^{a,b} = \frac{1}{|A|} \sum_{x \in A} \left|\mathsf{est}^{a,b}(x)\right|^2$. Observe that A, B are γ-good w.r. to f iff

$$\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right| < \gamma$$

By triangle $\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right|$ is at most

$$\left|\mathsf{est}^{a,b} - \frac{1}{|A|} \sum_{x \in A} \left|h^{a,b} * f(x)\right|^2\right| + \left|\frac{1}{|A|} \sum_{x \in A} \left|h^{a,b} * f(x)\right|^2 - \|h^{a,b} * f\|_2^2\right|$$

We bounds these two terms in the claims below, showing that for any $k > 1$ and $\eta > 0$, with probability at least $1 - \frac{1}{k} - 2\exp\left(-2|A|\eta^2/\|f\|_\infty^2\right)$,

$$\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right| \leq \eta(\eta + 2\|f\|_\infty) + 4k\|f\|_\infty^2 \exp\left(-\frac{2|B|\eta^2}{\|f\|_\infty^2}\right) + \eta$$

Assigning $k = 2/\delta$ and η, $|A|$, $|B|$ as in the lemma statement, we conclude that $\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right| \leq \gamma$ with probability at least $1 - \delta$.

Claim 3.21.1. *For any $x \in A$, $\left|\mathsf{est}^{a,b}(x) - h^{a,b} * f(x)\right| < \eta$, with probability at least $1 - 2\exp\left(-2|B|\eta^2/\|f\|_\infty^2\right)$.*

Proof. By definition of the convolution operator and of $h^{a,b}$,

$$h^{a,b} * f(x) = \mathop{\mathbb{E}}_{y \in \left\{0, \dots, 2^{\ell-1} - 1\right\}} \left[\chi_{-\lfloor (\frac{a+b}{2}) \rfloor}(y) f(x - y)\right]$$

Since $\mathsf{est}^{a,b}(x) = \frac{1}{|B|} \sum_{y \in B} \chi_{-\lfloor (\frac{a+b}{2}) \rfloor}(y) f(x-y)$ for B a random subset of $\left\{0, \dots, 2^{\ell-1} - 1\right\}$, then by Chernoff bound,

$$\left|\mathsf{est}^{a,b}(x) - h^{a,b} * f(x)\right| < \eta$$

with probability at least $1 - 2\exp\left(-2|B|\eta^2/\|f\|_\infty^2\right)$. □

Claim 3.21.2 (Bounding the first term)**.** *For any $k > 1$ and $\eta > 0$,*

$$\left|\mathsf{est}^{a,b} - \frac{1}{|A|}\sum_{x\in A}\left|h^{a,b} * f(x)\right|^2\right| \leq \eta(\eta + 2\|f\|_\infty) + 4k\|f\|_\infty^2 \exp\left(-\frac{2|B|\eta^2}{\|f\|_\infty^2}\right)$$

with probability at least $1 - \frac{1}{k}$

Proof. Fix some $k > 1$, $\eta > 0$ and $x \in A$. By Claim 3.21.1, $\left|\mathsf{est}^{a,b}(x) - h^{a,b} * f(x)\right| < \eta$, with probability at least $1 - 2\exp\left(-2|B|\eta^2/\|f\|_\infty^2\right)$.

Next we observe that $\left|\mathsf{est}^{a,b}(x) - h^{a,b} * f(x)\right| < \eta$ implies that

$$\left|\left|\mathsf{est}^{a,b}(x)\right|^2 - \left|h^{a,b} * f(x)\right|^2\right| \leq \eta(\eta + 2\|f\|_\infty)$$

This is by using the algebraic manipulations $|u^2 - v^2| = |(u-v)(u+v)| \leq |u - v|\left(|u - v| + 2|v|\right)$ on $u = \mathsf{est}^{a,b}(x)$ and $v = h^{a,b} * f(x)$ and recalling that by Item 4 in Lemma 3.31, $h^{a,b} * f(x) \leq \|f\|_\infty$.

We next show that with probability $1 - \frac{1}{k}$, $\left|\left|\mathsf{est}^{a,b}(x)\right|^2 - \left|h^{a,b} * f(x)\right|^2\right| \leq \eta(\eta + 2\|f\|_\infty)$ for at least $1 - k \cdot 2\exp\left(-2|B|\eta^2/\|f\|_\infty^2\right)$ of the $x \in A$. By the above, the expected number of $x \in A$ s.t. $\left|\left|\mathsf{est}^{a,b}(x)\right|^2 - \left|h^{a,b} * f(x)\right|^2\right| > \eta(\eta+2\|f\|_\infty)$ is at most $2\exp\left(-2|B|\eta^2/\|f\|_\infty^2\right)$. By Markov bound, with probability at most $\frac{1}{k}$, the number of such x's is k times its expectation.

Consider the case that indeed $\left|\left|\mathsf{est}^{a,b}(x)\right|^2 - \left|h^{a,b} * f(x)\right|^2\right| \leq \eta(\eta + 2\|f\|_\infty)$ for at least $1 - k \cdot 2\exp\left(-2|B|\eta^2/\|f\|_\infty^2\right)$ of the $x \in A$. Observe that for any $x \in A$, $\left|\left|\mathsf{est}^{a,b}(x)\right|^2 - \left|h^{a,b} * f(x)\right|^2\right| \leq 2\|f\|_\infty^2$. Putting it together we conclude that $\left|\mathsf{est}^{a,b} - \frac{1}{|A|}\sum_{x\in A}\left|h^{a,b} * f(x)\right|^2\right|$ is upper bounded by

$$\eta(\eta + 2\|f\|_\infty) \cdot \left(1 - 2k\exp\left(-\frac{2|B|\eta^2}{\|f\|_\infty^2}\right)\right) + 4k\|f\|_\infty^2 \exp\left(-\frac{2|B|\eta^2}{\|f\|_\infty^2}\right)$$

which is trivially upper bounded by $\eta(\eta + 2\|f\|_\infty) + 4k\|f\|_\infty^2 \exp\left(-\frac{2|B|\eta^2}{\|f\|_\infty^2}\right)$ □

Claim 3.21.3 (Bounding the second term)**.** *For every $\eta > 0$,*

$$\left|\frac{1}{|A|}\sum_{x\in A}\left|h^{a,b} * f(x)\right|^2 - \|h^{a,b} * f\|_2^2\right| \leq \eta$$

with probability at least $1 - 2\exp\left(-2|A|\eta^2/\|f\|_\infty^2\right)$.

Proof. The claim follows from Chernoff bound. □

□ □

3.3.2 Analysis of the SFT Algorithm over $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$

In this section we analyze the SFT Algorithm 3.9, that is, the SFT algorithm for complex functions f over $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$.

To prove the correctness of the SFT algorithm, we first define a filter function $h^{\alpha^t,a,b}$, and define a criterion on when the sets $A, B_{t,\ell}$, $t \in [k], \ell \in [\log N_t]$ chosen by the Generate Queries Algorithm 3.10 are good. We then analyze the SFT algorithm showing in Theorem 3.24 that the algorithm succeeds when the set $A, B_{t,\ell}$ chosen by the Generate Queries Algorithm 3.10 are good, and that these sets $A, B_{t,\ell}$ are good with high probability (where the probability is over the random coins of the Generate Queries Algorithm 3.10).

Notation. Let $G = \mathbb{Z}_{N_1}\times,\ldots,\times\mathbb{Z}_{N_k}$. For any $t \in [k]$ and $\alpha = (\alpha_1, \ldots, \alpha_k) \in G$, denote $\alpha^t = (\alpha_1, \ldots, \alpha_{t-1})$, $\bar{\alpha}^t = (\alpha_{t+1}, \ldots, \alpha_k)$.

The filter function $h_G^{\alpha^t,a,b}$ that we define is a phase shift of a periodic square function over the group $\mathbb{Z}_{N_t}$. The phase shift if by phase χ_β for $\beta = (\alpha^t, -\frac{b+a}{2}, 0, \ldots, 0)$ an element of G whose first $t-1$ coordinates are equal to α^t, whose t-th coordinate is $-s$ for s the center of the interval $\{a, \ldots, b\}$ and whose following $k-t$ coordinates are all zero. The function is normalized by the factor $\left(\prod_{i=t+1}^k N_i\right) \cdot \frac{N_t}{\lfloor (N_t/2(b-a)) \rfloor}$.

Definition 3.22 ($h_G^{\alpha^t,a,b}$)**.** *For every finite abelian group $G = \mathbb{Z}_{N_1}\times,\ldots,\times\mathbb{Z}_{N_k}$, an index $t \in [k]$, a vector $\alpha^t = (\alpha_1, \ldots, \alpha_{t-1})$ in $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$ and $a \le b$ in $\mathbb{Z}_{N_t}$, we define a filter function $h_G^{\alpha^t,a,b} \colon G \to \mathbb{C}$ by*

$$h_G^{\alpha^t,a,b}(y) \overset{def}{=} \begin{cases} \left(\prod_{i=t+1}^k N_i\right) \cdot \chi_{\alpha^t}(y^t) \cdot h_{N_t}^{a,b}(y_t) & \text{if } \bar{y}^t = 0^{k-t} \\ 0 & \text{otherwise} \end{cases}$$

for $h_{N_t}^{a,b}$ as in Definition 3.14 and $\chi_{\alpha^t}(y^t) = \prod_{j=1}^{t-1} e^{i\frac{2\pi}{N_j}\alpha_j y_j}$ a character in the group $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$. When G is clear from the context, we often omit it, and denote $h^{\alpha^t,a,b} = h_G^{\alpha^t,a,b}$.

We say that the sets $A, B_{t,\ell}$ chosen by the Generate Queries Algorithm 3.10 are *good*, if the values $\mathsf{est}^{\alpha^t,a,b}$ computed by the Distinguishing Algorithm 3.12 throughout the execution of the SFT algorithm all satisfy that

$$\mathsf{est}^{\alpha^t,a,b} \approx \|h^{\alpha^t,a,b} * f\|_2^2$$

for f the input function.

Definition 3.23 (Good A, B)**.** *For every finite abelian group $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, a function $f \colon G \to \mathbb{C}$, subsets $A, B \subseteq G$, an index $t \in [k]$, a vector $\alpha^t = (\alpha_1, \ldots, \alpha_{t-1})$ in $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$, an interval $\{a, \ldots, b\}$ in $\mathbb{Z}_{N_t}$, and a threshold $\gamma \in \mathbb{R}^+$, we say that A, B are γ-*good w.r. to *f, α^t, a, b, if*

$$\left|\mathsf{est}^{\alpha^t,a,b} - \|h^{\alpha^t,a,b} * f\|_2^2\right| < \gamma$$

for $\mathsf{est}^{\alpha^t,a,b} = \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B}\chi_{\alpha^t}(y^t)\chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y_t)\cdot f(x-y)\right)^2$ *the value computed by the Distinguishing Algorithm 3.12 on input* $\alpha^t, a, b, \tau, A, B$ *and* $\{(q, f(q))\}_{q\in A-B}$.

We say that a call to the Distinguishing Algorithm 3.12 with parameters α^t, $\{a, b\}$, τ, A, B, $\{(q, f(q))\}_{q\in A-B}$ *is* good *if* A, B *are* $\frac{\tau}{36}$*-good w.r. to* f, a^t, a, b.

Analyzing SFT Algorithm 3.9

In Theorem 3.24 below we show the SFT algorithm returns a short list containing all significant Fourier coefficients of f, and that its running time is polynomial in $\log|G|, \frac{1}{\tau}$.

The proof of Theorem 3.24 builds on three lemmata: In Lemma 3.28 we show that the sets $A, B_{t,\ell}$ chosen by the Generate Queries Algorithm 3.10 are good with high probability. In Lemma 3.26 we show that when $A, B_{t,\ell}$ are good then all calls to the Distinguishing Algorithm 3.12 are good. In Lemma 3.25 we show that when all calls to the Distinguishing Algorithm 3.12 are good, then the Fixed Queries SFT Algorithm 3.11 outputs a short list containing every significant Fourier coefficient of f. This in turn implies that the SFT Algorithm 3.9 outputs a short list containing every significant Fourier coefficient of f.

Theorem 3.24 (Analyzing Algorithm 3.9)**.** *Let the input to the SFT Algorithm 3.9 be a description* $\{(1, N_i)\}_{i=1}^k$ *of the group* $G = \mathbb{Z}_{N_1}\times\ldots\times\mathbb{Z}_{N_k}$, $\tau\in\mathbb{R}^+$, *and query access to a function* $f\colon G\to\mathbb{C}$*; denote its output by* L*. Then, with probability at least* $1-\delta'$*, the following hold:*

1. $L \supseteq \mathsf{Heavy}_\tau(f)$*, i.e.,* L *contains all* τ*-significant Fourier coefficients of* f
2. $|L| \le O(1/\tau)$*, i.e.,* L *is small, and*
3. *The running time of the SFT Algorithm 3.9 is at most polynomial in* $\log|G|, \frac{1}{\tau}$

Remark. Details on the running time of the SFT Algorithm 3.9 are as follows. For Boolean f the running time is at most $\widetilde{\Theta}\left(\frac{\log|G|}{\tau^{6.5}}\cdot\ln\frac{1}{\delta'}\right)$. For general functions f, the running time is at most $\widetilde{\Theta}\left(\log|G|\cdot\frac{1}{\tau}\cdot\left(\frac{\|f\|_2^2}{\tau}\right)^{1.5}\cdot\left(\frac{\|f\|_\infty^2}{\eta^2}\cdot\ln\frac{1}{\delta}\right)^2\right)$ for $\eta = \Theta\left(\min\left\{\tau, \sqrt{\tau}, \frac{\tau}{\|f\|_\infty}\right\}\right)$ and $\delta = \delta'/O\left(\frac{1}{\tau}\cdot\left(\frac{\|f\|_2^2}{\tau}\right)^{1.5}\log|G|\right)$.

Proof. Let δ be as set in step 1 of the SFT Algorithm 3.9. Below we show that all calls to the Distinguishing Algorithm in step 2(a)iiA of the Fixed Queries Algorithm 2(a)iiA are good with probability at least $1-\log|G|\cdot\Theta\left(\frac{1}{\tau}\cdot(\|f\|_2^2/\tau)^{1.5}\right)\cdot\delta$. Assigning $\delta = \delta'/\left(\log|G|\cdot\Theta\left(\frac{1}{\tau}\left(\|f\|_2^2/\tau\right)^{1.5}\right)\right)$ as was set in step 1 of the SFT Algorithm 3.9, we get that the all calls to the Distinguishing Algorithm 3.12 in step 2(a)iiA of the Fixed Queries Algorithm are good with probability at least $1-\delta'$. By Lemma 3.25 this concludes the proof.

In the following we abbreviate by saying "call" to refer to a call to the Distinguishing Algorithm 3.12 in step 2(a)iiA of the Fixed Queries Algorithm 3.11. By Lemma 3.28, each single call is good with probability at least $1-\delta$. We show that all calls are good with high probability using induction and union bound as follows. For $t=\ell=0$, $\left|Candidate_{\text{empty string},0}\right| = 1$, so there is a single call, an by the above it is good with probability at least $1-\delta$. Fix t,ℓ. Assuming all previous calls were good then $|Prefixed_{t-1}| \leq O(1/\tau)$ and $|Candidate_{\alpha^t,\ell}| \leq \Theta\left((\|f\|_2^2/\tau)^{1.5}\right)$. Therefore, if all calls were good, then the total number of calls is bounded by $\sum_{t=1}^{k}\sum_{\alpha^t\in Prefixes_t}\sum_{\ell=1}^{\log N_t}|Candidate_{\alpha^t,\ell}| \leq \log|G|\cdot\Theta\left(\frac{1}{\tau}(\|f\|_2^2/\tau)^{1.5}\right)$. By union bound all calls are good with probability at least $1-\log|G|\cdot\Theta\left(\frac{1}{\tau}(\|f\|_2^2/\tau)^{1.5}\right)\cdot\delta$. □

Analyzing Fixed Queries SFT Algorithm 3.11

In Lemma 3.25 below we show that when all calls to the Distinguishing Algorithm 3.12 are good, then the Fixed Queries SFT Algorithm 3.11 outputs a short list containing every significant Fourier coefficient of f.

Terminology. Let $t=0,\dots,k$ and $\alpha=(\alpha_1,\dots,\alpha_t)\in\mathbb{Z}_{N_1}\times\dots\times\mathbb{Z}_{N_t}$. For any $\beta=(\beta_1,\dots,\beta_k)\in\mathbb{Z}_{N_1}\times\dots\times\mathbb{Z}_{N_k}$, we say that β is *a completion of* α if $\beta_i=\alpha_i$ $\forall i=1,\dots,t$. We say that α is *a τ-significant t-prefix*, if there exists a τ-significant completion of α. Let $\alpha'=(\alpha'_1,\dots,\alpha'_{t+1})\in\mathbb{Z}_{N_1}\times\dots\times\mathbb{Z}_{N_{t+1}}$, we say that *$\alpha'$ is a continuation of α* if $\alpha'_i=\alpha_i$ $\forall i=1,\dots,t$.

Lemma 3.25 (Analyzing Fixed Queries SFT Algorithm 3.11)**.** *Let the input to the Fixed Queries Algorithm 3.6 be a description $\{(1,N_i)\}_{i=1}^k$ of the group $G=\mathbb{Z}_{N_1}\times\dots\times\mathbb{Z}_{N_k}$, $\tau\in\mathbb{R}^+$, $A,\{B_{t,\ell}\}_{t\in[k],\ell\in[\log N_t]}\subseteq G$ and $\{(q,f(q))\}_{q\in Q}$ for $Q=A-\bigcup_{t\in[k],\ell\in[\log N_t]}B_{t,\ell}$; denote its output by L.*

If all calls to the Distinguishing Algorithm 3.12 in step 2(a)i of the Fixed Queries Algorithm 3.11, are good, then the following holds:

1. *$L\supseteq \mathsf{Heavy}_\tau(f)$, i.e., L contains all τ-significant Fourier coefficients*
2. *$|L|\leq O(1/\tau)$, i.e., L is short, and*
3. *The running time of the Fixed Queries SFT Algorithm 3.11 is at most* $\Theta\left(\log|G|\cdot\frac{1}{\tau}\cdot\left(\frac{\|f\|_2^2}{\tau}\right)^{1.5}\cdot|Q|\right)$

Proof. We first show that for each $t\in[k]$, $Prefixes_t$ is a list of size at most $O(1/\tau)$ that contains all τ-significant t-prefixes of f. We prove by induction over t. At the first step of the algorithm $Prefixes_0=\{\text{empty string}\}$, which trivially satisfies the above two properties. Next we show that if the conditions of the theorem hold for $Prefixes_{t-1}$, then they also hold for $Prefixes_t$. By Lemma 3.25.1 below, for every $t\in[k]$ and $\alpha^t\in Prefixes_{t-1}$, $L_t(\alpha^t)$ contains all τ-significant continuations of α^t, moreover, all elements in $L(\alpha^t)$ are $O(\tau)$-significant t-prefixes. Since by induction assumption $Prefixes_{t-1}$ covers all τ-significant $(t-1)$-prefixes of f, then the following holds: (1) $Prefixes_t = \bigcup_{\alpha^t \in Prefixes_{t-1}} L_t(\alpha^t)$ covers all τ-significant t-prefixes of f, and (2) all elements if $Prefixes_t$ are $O(\tau)$ significant t-prefixes, implying that $|Prefixes_t| \leq O(1/\tau)$.

The running time is $T = \sum_{t=1}^{k} |Prefixes_t| \cdot T_{N_t}$ where T_{N_t} is the running time of the Fixed Queries SFT Algorithm for functions over $\mathbb{Z}_{N_t}$. Recall that $T_{N_t} \leq \Theta\left(\log N_t \cdot \left(\frac{\|f\|_2^2}{\tau}\right)^{1.5} \cdot |Q|\right)$ and $|Prefixes_t| \leq O(1/\tau)$. The running is therefore at most $\Theta\left(\sum_{t=1}^{k} \log N_t \cdot \frac{1}{\tau} \cdot \left(\frac{\|f\|_2^2}{\tau}\right)^{1.5} \cdot |Q|\right)$.

Lemma 3.25.1. *For any $t \in [k]$ and $\alpha^t \in Prefixes_{t-1}$, $L_t(\alpha^t)$ is a set that contains all τ-significant continuations of α^t, and each $\alpha \in L_t(\alpha^t)$ is a $\tau/2$-significant continuation of α^t.*

Proof. The proof is analogous to the proof of Lemma 3.17.1 while replacing Lemma 3.18 by Lemma 3.26. Details omitted. □ □

Analyzing Distinguishing Algorithm 3.12

In Lemma 3.26 we show that when the sets chosen by the Generate Queries Algorithm 3.10 are good, then the Distinguishing Algorithm keeps all intervals $\{a, b\}$ that contain a significant Fourier coefficient of f, and moreover, it keeps not too many intervals.

The proof of Lemma 3.26 relies on properties of the filter $h^{\alpha^t,a,b}$. These properties are stated in Lemma 3.27 below. The proof of these properties is deferred to section 3.4.3.

Lemma 3.26 (Analyzing Distinguishing Algorithm 3.12)**.** *Let the input to the Distinguishing Algorithm 3.12 be $\alpha^t \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$, $\{a, b\} \in \mathbb{Z}_{N_t} \times \mathbb{Z}_{N_t}$, $\tau \in \mathbb{R}^+$, $A, B \subseteq G$ s.t. that A, B are $\tau/36$-good w.r. to f, α^t, a, b, and $\{(q, f(q))\}_{q \in A-B}$; denote its output by "decision". Then the following holds:*

- *If $\exists \beta \in G$ s.t. $\beta^t = \alpha^t$, $\beta_t \in \{a, \ldots, b\}$ and $\left|\widehat{f}(\beta)\right|^2 \geq \tau$, then decision $= 1$*
- *Denote $\Delta = \sqrt{\frac{24}{\tau}}\|f\|_2$ and $Ext^{a,b} = \left\{\alpha \mid \mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \Delta \cdot 2(b-a)\right\}$. If decision $= 1$, then $\sum_{\beta \text{ s.t. } \beta^t = \alpha^t, \beta_t \in Ext^{a,b}} \left|\widehat{f}(\alpha)\right|^2 \geq \tau/12$*

Proof. The proof follows similarly to the proof of Lemma 3.18 while replacing the use of Lemma 3.19 regarding $h^{a,b}$ by use of Lemma 3.27 regarding $h^{\alpha^t,a,b}$.

Let us elaborate. Let $\mathsf{est}^{\alpha^t,a,b} = \frac{1}{|A|}\sum_{x \in A}\left(\frac{1}{|B_{t,\ell+1}|}\sum_{y \in B_{t,\ell+1}} \chi_{\alpha^t}(y^t)\chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y_t) f(x-y)\right)^2$ as defined in the Distinguishing Algorithm 3.12. Since A, B are $\tau/36$-good w.r. to f, α^t, a, b then

$$\left|\mathsf{est}^{\alpha^t,a,b} - \|h^{a,b} * f\|_2^2\right| < \tau/36$$

Recall that the Distinguishing Algorithm outputs *decision* $= 1$ iff $\mathsf{est}^{a,b} > \frac{5}{36}\tau$. By Lemma 3.27 when assigning $\gamma = \tau/36$ and $\lambda = \frac{5}{6}\tau$, the conditions of this lemma holds. Namely, if $\exists \beta \in G$ s.t. $\beta^t = \alpha^t$, $\beta_t \in \{a, \ldots, b\}$ and $\left|\widehat{f}(\beta)\right|^2 \geq \tau$, then $\mathsf{est}^{\alpha^t,a,b} \geq \frac{5}{36}\tau$; conversely, if $\mathsf{est}^{\alpha^t,a,b} \geq \frac{5}{36}\tau$, then $\sum_{\beta,\beta^t=\alpha^t,\beta_t \in Ext^{a,b}} \left|\widehat{f}(\beta)\right|^2 \geq \frac{1}{12}\tau$. □

Lemma 3.27 (Properties of $\|h^{\alpha^t,a,b} * f\|_2^2$)**.** *Let $f\colon \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k} \to \mathbb{C}$, $t \in [k]$, $\alpha^t \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$ and $\mathsf{est}^{\alpha^t,a,b} \in \mathbb{R}$. For any $\gamma > 0$, if $\left|\mathsf{est}^{a,b} - \|h^{\alpha^t,a,b} * f\|_2^2\right| \leq \gamma$, then the following holds:*

- $\mathsf{est}^{\alpha^t,a,b} \geq \sum_{\beta,\beta^t=\alpha^t,\beta_t \in \{a,\ldots,b\}} \frac{\left|\widehat{f}(\beta)\right|^2}{6} - \gamma$. *In particular, if $\exists \beta \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ s.t. $\beta^t = \alpha^t$, $\beta_t \in \{a, \ldots, b\}$ and $\left|\widehat{f}(\beta)\right|^2 \geq \tau$, then $\mathsf{est}^{\alpha^t,a,b} \geq \frac{\tau}{6} - \gamma$*

- *Denote $\Delta = \sqrt{\frac{2}{3\gamma}}\|f\|_2$ and $Ext^{a,b} = \left\{\alpha \mid \mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \Delta \cdot 2(b-a)\right\}$. For all $\lambda > 0$, if $\mathsf{est}^{\alpha^t,a,b} \geq \lambda$, then $\sum_{\beta \text{ s.t. } \beta^t=\alpha^t,\beta_t \in Ext^{a,b}} \left|\widehat{f}(\beta)\right|^2 \geq \lambda - 2\gamma$*

Proof. Proof appears in section 3.4.3. □

Analyzing Generate Queries Algorithm 3.10

In Lemma 3.28 we show that the sets $A, B_{t,\ell}$ chosen by the Generate Queries Algorithm 3.10 are good with high probability.

Lemma 3.28 (Analyzing Generate Queries Algorithm 3.5)**.** *Let the input to the Generate Queries Algorithm 3.5 be a description $\{(1, N_i)\}_{i=1}^k$ of the group $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, $\gamma \in \mathbb{R}^+$, $\|f\|_\infty$ and $\delta \in (0,1)$. Denote its output by A, $\{B_{t,\ell}\}_{t\in[k],\ell\in[\log N_t]}$ and $\{(q, f(q))\}_{q\in Q}$ for $Q = A - \bigcup_{t\in[k],\ell\in[\log N_t]} B_{t,\ell}$. Then for each $t \in [k]$, $\alpha^t \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$, $\ell \in \{0, \ldots, \log N_t\}$ and $a, b \in \mathbb{Z}_{N_t}$ s.t. $b - a = \lceil(N/2^\ell)\rceil$, $A, B_{t,\ell}$ are γ-good w.r. to f, α^t, a, b with probability at least $1 - \delta$.*

Proof. Fix $t \in [k]$, $\alpha^t \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$, $\ell \in \{0, \ldots, \log N_t\}$ and $a, b \in \mathbb{Z}_{N_t}$ s.t. $b - a = \lceil(N/2^\ell)\rceil$.

Denote $\mathsf{est}^{\alpha^t,a,b} = \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B} \chi_{\alpha^t}(y^t)\chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y)f(x-y)\right)^2$.

By Lemma 3.29 below, with γ, δ and $A, B_{t,\ell}$ as in Generate Samples Algorithm 3.10

$$\left|\mathsf{est}^{\alpha^t,a,b} - \|h^{\alpha^t,a,b} * f\|_2^2\right| < \gamma$$

with probability at least $1 - \delta$. Namely, $A, B_{t,\ell}$ are good w.r. to f, α^t, a, b with probability at least $1 - \delta$. □

Lemma 3.29. *Let $\gamma \in \mathbb{R}^+$, $\delta \in (0,1)$, $\alpha^t \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$, $\ell \in \{0, \ldots, \log N_t - 1\}$, $a, b \in \mathbb{Z}_{N_t}$ s.t. $b - a = \lceil(N_t/2^\ell)\rceil$ and $f\colon \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k} \to \mathbb{C}$. Let $A \subseteq \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ and $B \subseteq \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}} \times \{a, \ldots, b\} \times \{0\} \times \ldots \times \{0\}$ be random subsets.*

For any $\gamma > 0$ and $\delta \in (0,1)$, denote $\eta = O\left(\min\left\{\gamma, \sqrt{\gamma}, \frac{\gamma}{\|f\|_\infty}\right\}\right)$, if $|A| = \Theta\left(\left(\frac{\|f\|_\infty}{\eta}\right)^2 \ln\frac{1}{\delta}\right)$ and $|B| = \Theta\left(\left(\frac{\|f\|_\infty}{\eta}\right)^2 \ln\frac{\|f\|_\infty}{\delta\gamma}\right)$ are sufficiently large, then A, B are γ-good w.r. to f, α^t, a, b with probability at least $1-\delta$.

Remark. *In particular for Boolean f: for any $\gamma > 0$ and $\delta \in (0,1)$, if $|A| = \Theta\left(\frac{1}{\gamma^2}\ln\frac{1}{\delta}\right)$ and $|B| = \Theta\left(\frac{1}{\gamma^2}\ln\frac{1}{\delta\gamma}\right)$, then A, B are γ-good w.r. to f, α^t, a, b with probability at least $1-\delta$.*

Proof. The proof is similar to proof of Lemma 3.21. The only difference is that we replace Claim 3.21.1 with the claim below.

Claim 3.29.1. *For any $x \in A$, $\left|\mathsf{est}^{\alpha^t,a,b}(x) - h^{\alpha^t,a,b} * f(x)\right| < \eta$, with probability at least $1 - 2\exp\left(-2|B|\eta^2/\|f\|_\infty^2\right)$.*

Proof. By definition of the convolution operator and of $h^{\alpha^t,a,b}$,

$$h^{\alpha^t,a,b} * f(x) = \mathop{\mathbb{E}}_{y \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_t} \times \{0\} \times \ldots \times \{0\}} \left[\chi_{\alpha^t}(y^t) \cdot \chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y_t) \cdot f(x-y)\right]$$

Since $\mathsf{est}^{a,b}(x) = \frac{1}{|B|}\sum_{y \in B} \chi_{-\alpha^t}(y^t) \cdot \chi_{-\lfloor(\frac{a+b}{2})\rfloor}(y_t) \cdot f(x-y)$ for B a random subset of $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_t} \times \{0\} \times \ldots \times \{0\}$, then by Chernoff bound,

$$\left|\mathsf{est}^{\alpha^t,a,b}(x) - h^{\alpha^t,a,b} * f(x)\right| < \eta$$

with probability at least $1 - 2\exp\left(-2|B|\eta^2/\|f\|_\infty^2\right)$. □

□

3.3.3 Analysis of SFT Algorithm over Finite Abelian Groups

In this section we analyze the SFT Algorithm for complex functions f over arbitrary finite abelian groups.

Theorem 3.30 (Analyzing SFT Algorithm over Finite Abelian Groups)**.** *Let the input to the SFT Algorithm 3.9 be a description $\{(g_j, N_j)\}_{j=1}^k$ of the group G, $\tau \in \mathbb{R}^+$, and query access to a function $f\colon G \to \mathbb{C}$; denote its output by L. Then, with probability at least $1-\delta'$, the following hold:*

1. *$L \supseteq \mathsf{Heavy}_\tau(f)$, i.e., L contains all τ-significant Fourier coefficients of f*

2. *$|L| \le O(1/\tau)$, i.e., L is small, and*

3. *The running time of the SFT Algorithm is at most polynomial in $\log|G|, \frac{1}{\tau}$*

Proof. We first recap the description of the SFT algorithm for functions f over finite abelian groups G and then prove the theorem.

Let G be a finite abelian group of generators $g_1, \ldots, g_k$ with orders $N_1, \ldots, N_k$. The SFT algorithm utilizes the isomorphism between G and a direct product group $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ to operate as follows. It first finds the significant Fourier coefficients of the function f' over a direct product group defined by $f'(x_1, \ldots, x_k) = f(\prod_{j=1}^k g_j^{x_j})$ using the SFT Algorithm 3.9 and then maps the output list L' of τ-significant Fourier coefficients of f' to a list $L = \left\{ \prod_{j=1}^k g_j^{x_j} \,\middle|\, (x_1, \ldots, x_k) \in L' \right\}$ of elements in G.

To show that this SFT algorithm finds the τ-significant Fourier coefficients of the function $f \colon G \to \mathbb{C}$ it suffices to show that $\prod_{j=1}^k g_j^{x_j}$ is a τ-significant Fourier coefficient of f iff $(x_1, \ldots, x_k)$ is a τ-significant Fourier coefficient of f'. This follows immediately from the definition of the Fourier coefficients of f: Fix $(\alpha_1, \ldots, \alpha_k) \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, we show that $\widehat{f}(\prod_{j=1}^k g_j^{\alpha_j}) = \widehat{f'}(\alpha_1, \ldots, \alpha_k)$. The Fourier coefficient $\widehat{f}(\prod_{j=1}^k g_j^{\alpha_j})$ is:

$$\widehat{f}(\prod_{j=1}^k g_j^{\alpha_j}) = \frac{1}{|G|} \sum_{\prod_{j=1}^k g_j^{x_j} \in G} f(\prod_{j=1}^k g_j^{x_j}) \prod_{j=1}^k e^{i \frac{2\pi}{N_j} \alpha_j x_j}$$

The Fourier coefficient $\widehat{f'}(\alpha_1, \ldots, \alpha_k)$ is:

$$\widehat{f'}(\alpha_1, \ldots, \alpha_k) = \frac{1}{\prod_{j=1}^k N_j} \sum_{(x_1, \ldots, x_k) \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}} f'(x_1, \ldots, x_k) \prod_{j=1}^k e^{i \frac{2\pi}{N_j} \alpha_j x_j}$$

These two expressions are equal since $f'(x_1, \ldots, x_k) = f(\prod_{j=1}^k g_j^{x_j})$ and $|G| = \frac{1}{\prod_{j=1}^k N_j}$.

The fact that $|L| \leq O(1/\tau)$ follows immediately from Theorem 3.24 since $|L| = |L'|$ for L' the output of the SFT Algorithm 3.9.

The bound on the running time is immediate from the bound on the running time of the SFT Algorithm 3.9 proved in Theorem 3.24 together with the fact that query access to f' can be simulated in time $\log |G|$. $\square$

3.4 Properties of Filters $h^{a,b}$ and $h^{\alpha^t,a,b}$

In this section we prove the properties of the filter functions $h^{a,b}$ and $h^{\alpha^t,a,b}$ that we use in the analysis of the SFT algorithm. In section 3.4.1 we prove Lemma 3.19 on the properties of $h^{a,b}$. In section 3.4.2 we prove a technical proposition on characters sum which we use in the proof of Lemma 3.19. In section 3.4.3 we prove Lemma 3.27 on the properties of $h^{\alpha^t,a,b}$.

3.4.1 Properties of $h^{a,b}$: Proof of Lemma 3.19

In this section we present the proof of lemma 3.19. To ease the reading we first repeat the lemma below, and then present its proof.

Lemma 3.19. For any $\gamma > 0$, $f\colon \mathbb{Z}_N \to \mathbb{C}$ and $\mathsf{est}^{a,b} \in \mathbb{R}$, if $\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right| \leq \gamma$, then the following holds:

- $\mathsf{est}^{a,b} \geq \sum_{\alpha \in \{a,\dots,b\}} \frac{\left|\widehat{f}(\alpha)\right|^2}{6} - \gamma$. In particular, if $\exists \alpha_0 \in \{a,\dots,b\}$ with $\left|\widehat{f}(\alpha_0)\right|^2 \geq \tau$, then $\mathsf{est}^{a,b} \geq \frac{\tau}{6} - \gamma$
- For all $\lambda > 0$, if $\mathsf{est}^{a,b} \geq \lambda$, then $\sum_{\alpha \in Ext^{a,b}} \left|\widehat{f}(\alpha)\right|^2 \geq \lambda - 2\gamma$ for $Ext^{a,b} = \left\{\alpha \mid \mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \Delta \cdot 2(b-a)\right\}$ an extension of the interval $\{a,\dots,b\}$ defined with respect to a parameter $\Delta = \sqrt{\frac{2}{3\gamma}}\|f\|_2$.

Remark. We can strengthen the first part of the lemma to show that for any $\gamma' \in [0,1]$, $\mathsf{est}^{a,b} \geq \sum_{\mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \gamma' \frac{b-a}{2}} \left(1 - \frac{5}{6}(\gamma')^2\right) \left|\widehat{f}(\alpha)\right|^2 - \gamma$

Proof. We prove the first part of the lemma. By Parseval Identity and the Convolution-Multiplication Duality (see Theorem2.2 in Section 2.2),

$$\|h^{a,b} * f\|_2^2 = \sum_{\alpha \in \mathbb{Z}_N} \left|\widehat{h^{a,b}}(\alpha)\right|^2 \left|\widehat{f}(\alpha)\right|^2$$

which implies that $\|h^{a,b} * f\|_2^2 \geq \sum_{\alpha \in \{a,\dots,b\}} \left|\widehat{h^{a,b}}(\alpha)\right|^2 \left|\widehat{f}(\alpha)\right|^2$ which by Item 1 of Proposition 3.31, is at least as large as $\sum_{\alpha \in \{a,\dots,b\}} \frac{\left|\widehat{f}(\alpha)\right|^2}{6}$. Now, $\left|\mathsf{est}^{a,b} - \|h^{a,b} * f\|_2^2\right| \leq \gamma$ implies that

$$\mathsf{est}^{a,b} \geq \sum_{\alpha \in \{a,\dots,b\}} \frac{\left|\widehat{f}(\alpha)\right|^2}{6} - \gamma$$

Next, we prove the second part of the lemma. If $\mathsf{est}^{a,b} \geq \lambda$ then

$$\sum_{\alpha \in \mathbb{Z}_N} \left|\widehat{h^{a,b}}(\alpha)\right|^2 \left|\widehat{f}(\alpha)\right|^2 = \|h^{a,b} * f\|_2^2 \geq \lambda - \gamma$$

On the other hand,

$$\sum_{\alpha \notin Ext^{a,b}} \left|\widehat{h^{a,b}}(\alpha)\right|^2 \left|\widehat{f}(\alpha)\right|^2 \leq \left(\max_{\alpha \notin Ext^{a,b}} \left|\widehat{h^{a,b}}(\alpha)\right|^2\right) \cdot \sum_{\alpha \in \mathbb{Z}_N} \left|\widehat{f}(\alpha)\right|^2$$

which by Item 2 in Proposition 3.31 is smaller than

$$\frac{2}{3}\left(\frac{(b-a)}{\Delta \cdot 2(b-a)}\right)^2 \cdot \|f\|_2^2 = \gamma$$

(where the equality follows from the choice of Δ). Observe that we obtained a lower bound on the sum over all α's, and an upper bound on the sum over $\alpha \notin Ext^{a,b}$;

combining the two together we get that

$$\sum_{\alpha\in Ext^{a,b}} \left|\widehat{h^{a,b}}(\alpha)\right|^2 \left|\widehat{f}(\alpha)\right|^2 = \|h^{a,b} * f\|_2^2 - \sum_{\alpha\notin Ext^{a,b}} \left|\widehat{h^{a,b}}(\alpha)\right|^2 \left|\widehat{f}(\alpha)\right|^2 > (\lambda-\gamma)-\gamma$$

Finally, recall that by Item 3 of Proposition 3.31 $\left|\widehat{h^{a,b}}(\alpha)\right|^2 \leq 1$, therefore we conclude that

$$\sum_{\alpha\in Ext^{a,b}} \left|\widehat{f}(\alpha)\right|^2 > \lambda - 2\gamma$$

□

Proposition 3.31 (Properties of $h^{a,b}$)**.** *If $b-a \leq \frac{N}{2}$, then $h^{a,b}$ satisfies the following properties.*

1. *Pass Band: $\forall \alpha \in \mathbb{Z}_N$, if $\mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \gamma\frac{b-a}{2}$, then $\left|\widehat{h^{a,b}}(\alpha)\right|^2 > 1 - \frac{5}{6}\gamma^2$*

2. *Fast decreasing: $\forall \alpha \in \mathbb{Z}_N$, $\left|\widehat{h^{a,b}}(\alpha)\right|^2 < \left(\frac{2(b-a)}{\mathsf{abs}(\alpha - \frac{a+b}{2})}\right)^2$*

3. *Fourier bounded: $\forall \alpha \in \mathbb{Z}_N$, $\left|\widehat{h^{a,b}}(\alpha)\right|^2 \leq 1$*

4. *Convolution bounded: $\forall f \colon \mathbb{Z}_N \to \mathbb{C}$, $\|h^{a,b} * f\|_\infty \leq \|f\|_\infty$*

Moreover, if $a = b$, then $\widehat{h^{a,b}}(\alpha) = 1$ iff $\alpha = a$, and $\widehat{h^{a,b}}(\alpha) = 0$ otherwise.

Proof. By definition of the Fourier coefficient and of $h^{a,b}$, $\widehat{h^{a,b}}(\alpha) = \frac{1}{t}\sum_{y=0}^{t-1} \chi_{\alpha - \lfloor(\frac{a+b}{2})\rfloor}(y)$. Namely,

$$\widehat{h^{a,b}}(\alpha) = S_t\left(\alpha - \frac{a+b}{2}\right)$$

for S_t as in Definition 3.32. Items 1-3 therefore follow immediately from Proposition 3.33 in section 3.4.2.

We prove Item 4. By definition of the convolution operator,

$$\left|h^{a,b} * f(x)\right| = \left|\frac{1}{N}\sum_{y=0}^{N-1} h^{a,b}(y) f(x-y)\right|$$

By triangle inequality this is upper bounded by $\frac{1}{N}\sum_{y=0}^{N-1}\left|h^{a,b}(y)\right|\left|f(x-y)\right|$. By definition of $h^{a,b}$ this is upper bounded by $\frac{1}{t}\sum_{y=0}^{t-1}\left|f(x-y)\right|$ which is clearly upper bounded by $\|f\|_\infty$. □

3.4.2 Technical Proposition on Consecutive Characters Sum

We present a technical proposition on sums of consecutive characters. We use this proposition in the proof of Lemma 3.19.

Definition 3.32 ($S_t(\alpha)$)**.** *For any integer $t \in 1, \ldots, N$, let $S_t \colon \mathbb{Z}_N \to \mathbb{C}$ be the function defined by*

$$S_t(\alpha) \overset{def}{=} \frac{1}{t} \sum_{x=0}^{t-1} \chi_\alpha(x)$$

Proposition 3.33. *Let $t \in 1, \ldots, N$, and S_t as in the above definition, then the following properties hold.*

1. $|S_t(\alpha)|^2 = \frac{1-\cos(\frac{2\pi}{N}\alpha t)}{1-\cos(\frac{2\pi}{N}\alpha)}$

2. *Pass Band: $\forall \alpha \in \mathbb{Z}_N$ and $\gamma \in [0,1]$, if $\mathsf{abs}(\alpha) \leq \gamma \frac{N}{2t}$, then $|S_t(\alpha)|^2 > 1 - \frac{5}{6}\gamma^2$*

3. *Fast decreasing: $\forall \alpha \in \mathbb{Z}_N$, $|S_t(\alpha)|^2 < \frac{2}{3}\left(\frac{N/t}{\mathsf{abs}(\alpha)}\right)^2$*

4. *Fourier bounded: $\forall \alpha \in \mathbb{Z}_N$, $|S_t(\alpha)|^2 \leq 1$*

Proof. **Proof of Item 1.** Recall that $\chi_\alpha(x) = \omega^{\alpha x}$ for $\omega = e^{i\frac{2\pi}{N}}$ a primitive root of unity of order N. By the formula for geometric sum

$$S_t(\alpha) = \frac{1}{t}\frac{\omega^{-\alpha t} - 1}{\omega^{-\alpha} - 1}$$

Implying that

$$|S_t(\alpha)|^2 = \frac{1 - \cos(\frac{2\pi}{N}\alpha t)}{1 - \cos(\frac{2\pi}{N}\alpha)}$$

Proof of Item 2. For all $\alpha \in \mathbb{Z}_N$ with $\mathsf{abs}(\alpha) \leq \gamma\frac{N}{2t}$, we can utilizing Taylor approximation of the cosine function (namely, $1 - \frac{\theta^2}{2!} \leq \cos(\theta) \leq 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!}$) to have:

$$|S_t(\alpha)|^2 \geq 1 - \frac{\pi^2}{12}\left(\frac{2t\mathsf{abs}(\alpha)}{N}\right)^2 \geq 1 - \frac{\pi^2}{12}\gamma^2$$

and this is greater than $1 - \frac{5}{6}\gamma^2$ since $\pi^2 < 10$.

Proof of Item 3. As $\cos\theta = \cos(-\theta)$ and since $\mathsf{abs}(\alpha) \leq \frac{N}{2}$ we can, again, utilize Taylor approximation to have:

$$|S_t(\alpha)|^2 \leq \left(\frac{N/t}{\mathsf{abs}(\alpha)}\right)^2 \frac{1}{\pi^2\left(1 - \frac{\left(\frac{2\pi}{N}\mathsf{abs}(\alpha)\right)^2}{12}\right)} \leq \frac{2}{3}\left(\frac{N/t}{\mathsf{abs}(\alpha)}\right)^2$$

(where in the last inequality we used the bounds $\mathsf{abs}(\alpha) \leq N/2$ and $9 < \pi^2 < 10$).

Proof of Item 4. By triangle inequality, $|S_t(\alpha)| \leq \frac{1}{t}\sum_{x=0}^{t-1}|\chi_\alpha(x)|$ which is in turn equal to 1. $\square$

3.4.3 Properties of $h^{\alpha^t,a,b}$: Proof of Lemma 3.27

In this section we give the proof of lemma 3.27. To ease the reading we first repeat the lemma below, and then present its proof.

Lemma 3.27. Let $f\colon \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k} \to \mathbb{C}$, $t \in [k]$, $\alpha^t \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$ and $\mathsf{est}^{\alpha^t,a,b} \in \mathbb{R}$. For any $\gamma > 0$, if $\left|\mathsf{est}^{a,b} - \|h^{\alpha^t,a,b} * f\|_2^2\right| \leq \gamma$, then the following holds:

- $\mathsf{est}^{\alpha^t,a,b} \geq \sum_{\beta,\beta^t=\alpha^t,\beta_t\in\{a,\ldots,b\}} \frac{\left|\widehat{f}(\beta)\right|^2}{6} - \gamma$. In particular, if $\exists \beta \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ s.t. $\beta^t = \alpha^t$, $\beta_t \in \{a, \ldots, b\}$ and $\left|\widehat{f}(\beta)\right|^2 \geq \tau$, then $\mathsf{est}^{\alpha^t,a,b} \geq \frac{\tau}{6} - \gamma$

- Denote $\Delta = \sqrt{\frac{2}{3\gamma}}\|f\|_2$ and $Ext^{a,b} = \left\{\alpha \mid \mathsf{abs}(\alpha - \frac{a+b}{2}) \leq \Delta \cdot 2(b-a)\right\}$. For all $\lambda > 0$, if $\mathsf{est}^{\alpha^t,a,b} \geq \lambda$, then $\sum_{\beta \text{ s.t. } \beta^t=\alpha^t,\beta_t\in Ext^{a,b}} \left|\widehat{f}(\beta)\right|^2 \geq \lambda - 2\gamma$

Proof. The proof follows from Lemma 3.19 and Proposition 3.34. □

Proposition 3.34 (Properties of $h^{\alpha^t,a,b}$). $\|h^{\alpha^t,a,b} * f\|_2^2 = \sum_{\beta,\beta^t=\alpha^t} \left|\widehat{h^{a,b}}(\beta)\right|^2 \left|\widehat{f}(\beta)\right|^2$

Proof. By Parseval Identity (see Theorem 2.2), $\|h^{\alpha^t,a,b} * f\|_2^2 = \sum_\beta \left|\widehat{h^{\alpha^t,a,b} * f}(\beta)\right|^2$. By the Convolution-Multiplication Duality (see Theorem 2.2), this is equal to $\sum_\beta \left|\widehat{h^{\alpha^t,a,b}}(\beta)\right|^2 \left|\widehat{f}(\beta)\right|^2$. By the claim below $\widehat{h^{\alpha^t,a,b}}(\beta) = 0$ if $\alpha^t \neq \beta^t$ and $\widehat{h^{a,b}}(\beta_t)$ otherwise. Therefore we conclude that $\|h^{\alpha^t,a,b} * f\|_2^2 = \sum_{\beta \text{ s.t. } \beta^t=\alpha^t} \left|\widehat{h^{a,b}}(\beta)\right|^2 \left|\widehat{f}(\beta)\right|^2$.

Claim 3.34.1. *For all $\beta \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$,*

$$\widehat{h^{\alpha^t,a,b}}(\beta) = \begin{cases} \widehat{h^{a,b}}(\beta_t) & \text{if } \alpha^t = \beta^t \\ 0 & \text{otherwise} \end{cases}$$

Proof. By definition of Fourier transform, $\widehat{h^{\alpha^t,a,b}}(\beta) = \frac{1}{|G|}\sum_{y\in G} h^{\alpha^t,a,b}(y)\chi_{-\beta}(y)$. By definition of $h^{\alpha^t,a,b}$ we get that

$$\widehat{h^{\alpha^t,a,b}}(\beta) = \frac{1}{\prod_{i=1}^t N_i} \sum_{y\in G,\bar{y}^t=0^{k-t}} \chi_{\alpha^t}(y^t) h^{a,b}(y_t)\chi_{-\beta}(y)$$

Using the fact that $\chi_{-\beta}(y) = \chi_{-\beta^t}(y^t)\cdot\chi_{-\beta_{t+1}}(y_{t+1})\cdot\chi_{-\bar{\beta}^{t+1}}(\bar{y}^{t+1})$ and that $\chi_{-\bar{\beta}^t}(0^{k-t}) = 1$, we get that

$$\widehat{h^{\alpha^t,a,b}}(\beta) = \frac{1}{\prod_{i=1}^t N_i} \sum_{y^t\in\mathbb{Z}_{N_1},\ldots,\mathbb{Z}_{N_{t-1}}} \left(\chi_{\alpha^t}(y^t)\chi_{-\beta^t}(y^t) \sum_{y_t\in\mathbb{Z}_{N_t}} h^{a,b}(y_t)\chi_{-\beta_t}(y_t) \right)$$

By definition of the Fourier transform of $h^{a,b}$, $\frac{1}{N_t}\sum_{y_t \in \mathbb{Z}_{N_t}} h^{a,b}(y_t)\chi_{-\beta_t}(y_t) = \widehat{h^{a,b}}(\beta_t)$, thus,

$$\widehat{h^{\alpha^t,a,b}}(\beta) = \widehat{h^{a,b}}(\beta_t) \cdot \frac{1}{\prod_{i=1}^{t-1} N_i} \sum_{y^t \in \mathbb{Z}_{N_1},\ldots,\mathbb{Z}_{N_{t-1}}} \chi_{\alpha^t}(y^t)\chi_{-\beta^t}(y^t)$$

Finally, since $\frac{1}{\prod_{i=1}^{t-1} N_i}\sum_{y^t \in \mathbb{Z}_{N_1},\ldots,\mathbb{Z}_{N_{t-1}}} \chi_{\alpha^t}(y^t)\chi_{-\beta^t}(y^t) = \frac{1}{\prod_{i=1}^{t}}\sum_{y^t \in \mathbb{Z}_{N_1},\ldots,\mathbb{Z}_{N_{t-1}}} \chi_{\alpha^t-\beta^t}(y^t)$ is 1 iff $\alpha^t = \beta^t$ and is 0 otherwise, we conclude that

$$\widehat{h^{\alpha^t,a,b}}(\beta) = \begin{cases} \widehat{h^{a,b}(\beta)} & \text{if } \alpha^t = \beta^t \\ 0 & \text{otherwise} \end{cases}$$

□ □

Chapter 4

Learning Characters with Noise in Random Samples Access Model

In this chapter we study the problem of Learning Characters with Noise in the *random samples access model* (rLCN). We investigate definitional issues regarding rLCN studying various noise models by which the input character may be corrupted. We examine the tractability of rLCN in the various noise models, showing it is tractable in some and conjecturing it is intractable in others. For rLCN in the conjectured intractable noise models, we explore average vs. worst case relations showing it is random self reducible. Finally, we present a (non polynomial time) algorithm for rLCN in the conjectured intractable noise models.

4.1 Introduction

In this chapter we define and study the problem of Learning Characters with Noise in the random samples access model (rLCN, in short). In the rLCN problem the algorithm is given random samples access to a complex function $f\colon G \to \mathbb{C}$ over a finite abelian group G and the goal is to find characters of G that are close to f. By *random samples access* we mean that the algorithm cannot choose the specific values of f it sees; instead the algorithm has access to values $(x, f(x))$ for x chosen independently and uniformly at random from G. The *closeness* of f to a character may be measured by various models, *e.g.*, Hamming distance, ℓ_∞-distance and ℓ_2-distance.

To allow a broad array of models by which closeness of f to a character χ is measured, it is convenient to think of f as being generated by adding *noise* to χ, where the noise may be added by any algorithm. Each noise adding algorithm defines some noise model; thus allowing a broad spectrum of noise models. We say that "*f is a noisy α-character in noise model $\mathcal{S}$*" if f was obtained by adding noise to the character χ_α in noise model (*i.e.*, algorithm) $\mathcal{S}$. With this terminology, the goal of rLCN is to find the character χ_α when given a noisy α-character f.

Definition 4.1 (Random samples access)**.** *Let $f\colon G \to \mathbb{C}$. We say that an algorithm is given* random samples access *to f, if it can request and receive in unit time a pair*

$(x, f(x))$ *where x is drawn independently and uniformly at random from G.*

Definition 4.2 (rLCN in noise model $\mathcal{S}$)**.** *The input to* rLCN in noise model $\mathcal{S}$ *is a description of a finite abelian group G by its generators and their orders and random samples access to a function $f\colon G \to \mathbb{C}$ which is a noisy α-character in noise model $\mathcal{S}$; the goal is to find α.*

Considering the random samples access model is motivated by applications where query access is unattainable, and yet we are interested in finding characters close to a given function. Such applications arise in diverse areas, including data analysis, error correcting codes and cryptography; examples follow.

> **rLCN & data analysis.** An algorithm solving rLCN could be useful in data analysis tasks. Consider for example a biological study of gene expression patterns, where the input is a pool of cells, each labeled "sick" or "healthy" according to the tissue from which it was taken. Each cell exhibits some gene expression pattern modeled as a vector $p \in \mathbb{Z}_N^k$ for N denoting the number of measured expression levels and k the number of examined genes. We identify the data with a Boolean function over $\mathbb{Z}_N^k$, and focus on cases where analyzing the Fourier characters close to f is of interest. Observe that it is feasible to sample $(x, f(x))$ for a random gene expression pattern x out of the patterns in the studied cells. This is by sampling a random cell in the pool, analyzing its gene expression pattern x and reading its label $f(x)$. In contrast, finding the value $f(x)$ for a specific pattern x is by far more demanding, requiring analysis of practically all cells in the data to find one with the desired gene expression pattern (if exists).
>
> **rLCN & error correcting codes.** We show that an algorithm solving rLCN (in ℓ_2-distance while seeing $O(\log N)$ samples) would be useful for an error correcting tasks. Specifically, it would allow us to efficiently decode the family of binary codes of constant rate and distance presented in chapter 6.
>
> **rLCN & cryptography.** We show that an algorithm solving rLCN (in ℓ_2-distance) would be useful for solving a long standing open problem in cryptography. Specifically, it would allow proving for several deterministic predicates (*e.g.*, the most significant bit) that they are hardcore predicates for the Diffie-Hellman function. This would be a breakthrough result as there is no known deterministic hardcore predicate for the Diffie-Hellman function.
>
> We remark that conversely, if rLCN is intractable, this could also be useful from a cryptographic point of view. In modern cryptography, cryptographic protocols are based on computational intractability assumption. Existing assumptions come from Number Theory (*e.g.*, RSA, Discrete-Log, Diffie-Hellman), Geometry of Numbers (*e.g.*, finding the shortest vector in a lattice) and Computational Learning (*e.g.*, Learning Parity

with Noise). It is desirable to have cryptographic protocols based on versatile assumptions in case some assumptions prove to be false. Should rLCN be intractable, it could present a new candidate to base cryptographic protocols on.

For functions over the Boolean cube $\{0,1\}^n$ the problem of learning characters with noise in random samples access model has been studied extensively under the name *Learning Parity with Noise (LPN)*.[1] LPN has been studied in settings where there is no noise and in the Hamming noise model (adversarial or random). In the no-noise settings, LPN is known to be tractable by Gauss Elimination. In the Hamming noise model, LPN is believed to be intractable. This intractability conjecture is often referred to as the intractability of *decoding random linear codes*. LPN is random self reducible, that is, worst case instances are as hard as random ones (within the same input sizes). Feldman *et.al.* [30] consider another aspect of average and worst case relation, showing equivalence of LPN in average case *noise*, that is, random noise, and in worst case noise. Blum-Kalai-Wasserman [15] gave an algorithm solving rLPN in time $2^{O(n/\log n)}$. LPN has applications analogous to some of the applications we consider for rLCN: Tractability of LPN (when seeing $O(n)$ samples) would imply an efficient decoding algorithm for random linear codes. Conversely, intractability of LPN is employed in cryptographic protocols, including the McEliece cryptosystem [66] and signature scheme [24], the Hopper-Blum authentication protocol [55] and subsequent works regarding efficient authentication protocols [88].

For functions over product groups $\{0,1,\ldots,p\}^n$ of small prime characteristic $p \leq poly(n)$, Regev [74] suggested a generalization of LPN, conjectured it is intractable in the Gaussian noise model, and presented a cryptosystem based on this conjecture.

In this work, we focus our study on functions $f\colon \mathbb{Z}_N \to \mathbb{C}$ over cyclic groups $\mathbb{Z}_N$ of large characteristic N. We measure complexity in terms of the binary representation length of elements in $\mathbb{Z}_N$, that is, an efficient algorithm has running time polynomial in $\log N$.

We study rLCN in a variety of noise models, including models where noise is bounded by some distance measure (including: Hamming distance, ℓ_∞-distance, stochastic versions of the Hamming and ℓ_∞ distances, ℓ_2-distance and Boolean ℓ_2-distance),[2] as well as noise models designed to simulate product group structure in cyclic groups $\mathbb{Z}_N$ with N of small prime factors.

[1] The term *parity* emanates from the fact that the characters $\chi_{(\alpha_1,\ldots,\alpha_n)}(x_1,\ldots,x_k)$ of the Boolean cube $\{0,1\}^n$ measure the parity of the number of $i \in [n]$ s.t. both $\alpha_i = 1$ and $x_i = 1$.

[2] Let $f\colon G \to \mathbb{C}$ be a function, and $\chi_\alpha\colon G \to \mathbb{C}$ a character of the group G. We say that f is a noisy α-character in the *Hamming noise model of parameter* ε, if $f(x) \neq \chi_\alpha(x)$ for at most ε fraction of entries $x \in G$. We say that f is a noisy α-character in the *a stochastic version of Hamming noise model of parameter* ε, if $f(x) = \chi_\alpha(x)$ with probability $1-\varepsilon$ independently at random for each $x \in G$, and $f(x)$ may be chosen adversarially or at random otherwise. We say that f is a noisy α-character in the *ℓ_∞-noise model of parameter* ε, if $\|f-\chi_\alpha\|_\infty \leq \varepsilon$. We say that f is a noisy α-character in a *stochastic version of the ℓ_∞ noise model of parameters* ε, σ^2, if $f(x) - \chi_\alpha(x)$ is a random variable with expectation ε and variance σ^2. We say that f is a noisy α-character in the *ℓ_2 noise model of parameter* ε if $\|f-\chi\|_2^2 \leq \varepsilon$. We say that f is a noisy α-character in the *Boolean ℓ_2 noise model of parameter* ε if f is a Boolean function, and $\|f-\chi\|_2^2 \leq \varepsilon$.

Studying rLCN in a variety of noise models is motivated by two goals. On the positive side, we'd like to find efficient algorithms for rLCN in any noise model that may arise in practice. Thus, we are interested in proving *tractability* of rLCN in a variety of noise models. On the negative side, we'd like to find a noise model where rLCN is *intractable*. Such an intractability result could potentially be used in cryptography as an alternative to known functions on which cryptographic protocols are based.

Our Results

On the positive side, we show that rLCN over groups[3] $\{\mathbb{Z}_N\}_N$ is *tractable* in Hamming noise of parameter $\varepsilon \in [0, 1)$, and in the ℓ_∞ noise of parameter ε poly-logarithmically related to the group size. Similar results hold for the stochastic versions of the Hamming and ℓ_∞ noise models. We remark that in the Hamming noise case, when $\varepsilon < \frac{1}{2}$, the algorithm can return the *unique* closest character in running time polynomial in $\log N$ and $1/(\frac{1}{2} - \varepsilon)$, whereas in general the algorithm returns the *list* of $O(1/\varepsilon)$ closest characters in running time polynomial in $\log N$ and $1/(1 - \varepsilon)$.

On the negative side, we *conjecture* that rLCN is *intractable* in the ℓ_2 noise model and the Boolean ℓ_2 noise model. We bring supporting evidence to this conjecture relating it to applications in cryptography and coding theory. Specifically, we show that if this conjecture is false, then we can (1) decode random (non-linear) binary codes of constant rate and distance, and (2) present the first deterministic hardcore predicate for the Diffie-Hellman function.

Furthermore, we show that in the (Boolean) ℓ_2 noise model, rLCN is *random self reducible*. That is, in these noise models, if there is no algorithm solving rLCN on the worst case, then there is no algorithm solving rLCN on the average case (within the same input sizes).

We present a (non-polynomial time) algorithm for rLCN in (Boolean) ℓ_2 noise, running in time $N^{1-O(1/(\log\log N)^2)}$ for N the size of the domain of f. This gives a slight improvement over the $O(N \log N)$ running time of the naive algorithm.[4]

Finally, we investigate the correspondence between learning characters with noise for functions over *cyclic groups* and functions over *product groups*. We observe that for $N = p_1 p_2 \dots p_t$ a product of distinct primes, the cyclic group $\mathbb{Z}_N$ is isomorphic to the product group $\mathbb{Z}_{p_1} \times \dots \times \mathbb{Z}_{p_t}$. We then define noise models for functions over $\mathbb{Z}_N$ that simulate noise models for functions over product groups $\mathbb{Z}_{p_1} \times \dots \times \mathbb{Z}_{p_t}$. We explore whether the complexity of rLCN in those noise models is similar to its complexity for functions over product groups, showing it is not. Instead, the tractability/intractability of rLCN in these noise models depends on properties of the factorization of N.

[3]We say *rLCN over groups* $\{\mathbb{Z}_N\}_N$ to refer to rLCN when the input functions is function $f \colon \mathbb{Z}_N \to \mathbb{C}$ over the group $\mathbb{Z}_N$.

[4]The naive algorithm is an algorithm measuring correlation of $O(\log N)$ random samples with each character in G, outputting the character with highest correlation.

Chapter Organization

In section 4.2 we give an overview of our definitions and results. In section 4.3 we give the proofs of our tractability results. In section 4.4 we give the proof of our random self reducible result. In section 4.5 we present and analyze our (non polynomial time) algorithm for solving rLCN in the (Boolean) ℓ_2 noise model.

4.2 Overview of Definitions and Results

In this section we give an overview of our definitions and results regarding the problem of Learning Characters with Noise in random samples access model. That is, the problem of finding α, when given random samples access to a function f which is a noisy α-character in some noise model $\mathcal{S}$.

We focus on the case that f is a function over the group $\mathbb{Z}_N$ of integers modulo N for N a large number. We measure complexity in terms of the binary representation length $\log N$ of elements in $\mathbb{Z}_N$. We use the terminology *rLCN over* $\{\mathbb{Z}_N\}_{N\in\mathcal{I}}$ when we want to address rLCN on input restricted to functions over groups $\mathbb{Z}_N$ with N in the (possibly infinite) set $\mathcal{I}\subseteq\mathbb{N}$.

We consider various noise models $\mathcal{S}$ by which the noisy α-character f is generated. These noise models fall into two categories: "noise of bounded distance" where the distance of f from χ_α is bounded by some distance measure (*e.g.*, Hamming, ℓ_∞, and ℓ_2 distances), and "noise simulating product group structure" which is defined in correspondence to factorization of N into prime numbers.

In the following we first define the bounded distance noise models we consider and survey our investigation of rLCN in these noise models (in section 4.2.1). We then define noise models simulating product group structure and survey our results for rLCN in these noise models (in section 4.2.2).

4.2.1 rLCN in Bounded Distance Noise Models

We define the bounded distance noise models we consider, and survey our results on rLCN in these noise models.

Definition 4.3 (Bounded distance noise models). *Let $f\colon G\to\mathbb{C}$ be a complex function over a finite abelian group G, and let $\chi_\alpha\colon G\to\mathbb{C}$ be a character of the group G.*

- **Hamming noise:** *We say that f is a noisy α-character in the* Hamming noise model of parameter ε, *if $f(x)\neq\chi_\alpha(x)$ for at most ε fraction of entries $x\in G$.*

  We say that f is a noisy α-character in the a stochastic version of Hamming noise model of parameter ε, *if $f(x)=\chi_\alpha(x)$ with probability $1-\varepsilon$ independently at random for each $x\in G$.*

- ℓ_∞ **noise:** *We say that f is a noisy α-character in the* ℓ_∞-noise model of parameter ε, *if $\|f-\chi_\alpha\|_\infty\leq\varepsilon$.*

We say that f is a noisy α-character in a stochastic version of the ℓ_∞ noise model of parameters ε, σ^2, *if $f(x) - \chi_\alpha(x)$ is a random variable with expectation ε and variance σ^2.*

- ℓ_2 **noise:** *We say that f is a noisy α-character in the* ℓ_2 noise model of parameter ε *if* $\|f - \chi\|_2^2 \leq \varepsilon$.

  We say that f is a noisy α-character in the Boolean ℓ_2 noise model of parameter ε *if f is a Boolean function, and* $\|f - \chi\|_2^2 \leq \varepsilon$.

- **No noise:** *We use the terminology f is a noisy α-character in the* no noise model, *if* $f = \chi_\alpha$.

Remark 4.4. 1. *In all above noise models, noise may be* adversarial *where worst case f is chosen, or* random.

2. *We assume without lost of generality that $f(x) \in \{\chi_\alpha(x)\}_{x \in G}$ $\forall x$. Otherwise, we can (efficiently) round each $f(x)$ to its closest value there.*

3. *Stochastic ℓ_∞ noise model with parameter $\sigma^2 = 0$ is identical to the ℓ_∞ noise model.*

4. *If f is a noisy α-character in the ℓ_2 noise model of parameter ε, then f is positively correlated with χ_α, specifically, $\langle f, \chi_\alpha \rangle > 1 - \frac{\varepsilon}{2} + \frac{1}{2}(1 - \|f\|_2^2)$. In particular, for Boolean f, $\langle f, \chi_\alpha \rangle \geq 1 - \frac{\varepsilon}{2}$.*

5. *Let $\{\psi_{G,\alpha}\}_{G, \alpha \in G}$ be a family of functions defined over finite abelian groups G. The above noise models can be applied to $\psi_{G,\alpha}$ just as they were applied to the characters.*

Tractability Results for Bounded Distance Noise

We investigate the computational difficulty of rLCN in bounded distance noise models. We prove that rLCN is tractable under some noise models, and conjecture it is intractable for others.

Our tractability statements refer to rLCN over $\{\mathbb{Z}_N\}_{N \in \mathbb{N}}$ (that is, on inputs restricted to noisy characters f over groups $\mathbb{Z}_N$). We do not believe that our tractability results extend to general abelian groups, because, in particular, this would contradict the assumed intractability of the the problem of Learning Parity with Noise in random samples access model.

We show that rLCN is tractable in the noise models of: no noise, Hamming noise, ℓ_∞ noise and stochastic versions of Hamming and ℓ_∞ noise models.

Theorem 4.5. *rLCN over $\{\mathbb{Z}_N\}_{N \in \mathbb{N}}$ is in BPP in each of the following noise models:*

- *No noise*

- *Hamming noise and stochastic Hamming noise with parameter $\varepsilon \in [0,1)$.*

  When $\varepsilon < \frac{1}{2}$, the algorithm can return the unique *closest character in running time polynomial in* $\log N$ *and* $1/(\frac{1}{2} - \varepsilon)$. *In general, the algorithm returns the* list *of $O(1/\varepsilon)$ closest characters in running time polynomial in* $\log N$ *and* $1/(1-\varepsilon)$.

- *ℓ_∞ noise with parameter $\varepsilon \leq \sin(\frac{2\pi}{N} poly \log N)$, and stochastic ℓ_∞ noise with parameters $\varepsilon, \sigma \leq \sin(\frac{2\pi}{N} poly \log N)$*

Intractability Conjecture for Boolean ℓ_2 Noise

We conjecture the rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ is intractable in the ℓ_2 noise model. Furthermore, we conjecture it is intractable even for the restricted class of Boolean functions, that is, in the Boolean ℓ_2 noise model.

Conjecture 4.6. *rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ in ℓ_2 noise model is intractable. Moreover, rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ is intractable even in the* Boolean *ℓ_2 noise model.*

This conjecture rises from our experience with rLCN and from analogies between rLCN and the believed-to-be hard rLPN. Two of our results can be viewed as providing supporting evidence for this conjecture. We stress however that these results are by no mean the reason we make this conjecture. We elaborate on these results.

rLCN in Boolean ℓ_2 Noise vs. Decoding Random (non-linear) Codes Let $\mathcal{C}^{const}$ be the family of binary constant rate (non linear) codes define in Example 6.59 (in section 6.6 of chapter 6). We show that if there is an efficient algorithm solving rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ in Boolean ℓ_2 noise model, then a random code in $\mathcal{C}^{const}$ is efficiently decodable.

There are strong analogies between the code $\mathcal{C}^{const}$ and random linear codes. Both codes are defined by taking a random subset S of a group G ($G = \mathbb{Z}_N$ for $\mathcal{C}^{const}$, and $G = \{0,1\}^n$ for random linear codes), and encoding elements x in the group by a restriction of the character χ_x to the set S. (More accurately, a restriction of a *boolean-ized* version of χ_x, in the case of $\mathcal{C}^{const}$.) Both codes are binary and of constant rate and distance. It is widely believed that random linear codes are not efficiently decodable. By way of analogy it may very well be that random constant rate Multiplication Code are also not efficiently decodable.

Theorem 4.7. *Let $\mathcal{C}$ be the random code from Example 6.59. If there is an efficient algorithm solving rLCN over $\mathbb{Z}_N$ with $O(\log N)$ samples and with error probability poly$(\frac{1}{N})$, then $\mathcal{C}$ is efficiently decodable, with probability at least $2/3$.*
(By "efficient algorithm" we refer to algorithms with running time poly$(\log N)$.)

Proof. See details in Theorem 6.37 in section 6.2.5. $\square$

rLCN in Boolean ℓ_2 Noise vs. Diffie-Hellman Hardcore Predicates We relate rLCN to the question of proving deterministic hardcore predicates for the Diffie-Hellman function.

Let us briefly present the relevant definitions. The Diffie-Hellman function $DH = DH_{p,g} \colon \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ for p a prime and g a generator of $\mathbb{Z}_p^*$ is defined by $DH(g^a, g^b) = g^{ab} \mod p$. The Computational Diffie-Hellman (CDH) assumption asserts that DH is intractable (when considering the asymptotic family of $DH_{p,g}$ functions with growing p). The best known algorithm solving DH runs in time $O\left(\exp\left(O(\log n)^{1/3}(\log\log n)^{2/3}\right)\right)$ for $n = \log p$ the bit representation length of elements in $\mathbb{Z}_p$. For a predicate $P \colon \mathbb{Z}_p \to \{\pm 1\}$, we say that and algorithm B *predicts P w.r. to DH* if the probability that $B(g^a, g^b) = P(DH(g^a, g^b))$ has non-negligible advantage over a random guess. We say that P is hardcore predicate for DH, if given an algorithm B that predicts P w.r. to DH, there is an algorithm A that computes DH with a non-negligible success probability (where the probability is taken over the inputs to the inversion algorithm A and over its random coins); and the running time of A is $T_B \cdot poly \log N$ for T_B the running time of B.

We show that if rLCN over $\{\mathbb{Z}_p\}_{\text{prime } p}$ is in BPP, then every segment predicate (see Definition 7.8) is hardcore for DH. In particular, this implies the most significant bit of g^{ab} (*i.e.*, whether it is smaller or greater than $p/2$) is hardcore for DH.

Finding *any* hardcore predicate for DH is a long standing open problem (see a detailed discussion in section 7.5 in chapter 7). This leads to viewing this result as evidence on the intractability of rLCN over $\{\mathbb{Z}_p\}_{\text{prime } p}$.

We stress however that our result relating rLCN and DH hardcore predicates, albeit providing evidence for the hardness of rLCN, is by no mean the reason we believe rLCN to be hard. In fact, we believe rLCN to be *harder* than the problem of proving hardcore predicates for DH. Specifically, given a predictor algorithm B, DH can be solved in time $O\left(\exp\left(O(\log n)^{1/3}(\log\log n)^{2/3}\right)\right)$ (for $n = \log p$ the bit representation length of elements in $\mathbb{Z}_p$) by ignoring B and using the best known algorithm for computing DH. Our result relating rLCN and DH cannot therefore show that rLCN is harder to compute in time better than $O\left(\exp\left(O(\log n)^{1/3}(\log\log n)^{2/3}\right)\right)$. Whereas, we believe rLCN to be even harder than that.

Theorem 4.8. *If rLCN over $\{\mathbb{Z}_p\}_{\text{prime } p}$ in Boolean ℓ_2 noise model is in BPP, then every segment predicate is hardcore for the Diffie-Hellman function.*

Proof. See details in section 7.5 in chapter 7. □

Random Self Reducibility in ℓ_2 and Boolean ℓ_2 Noise

We explore the average-case vs. worst-case hardness of rLCN. Recall that a problem is *random self reducible* if solving it on the worst case is efficiently reducible to solving it on the average case. In the case of rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ in ℓ_2 noise we show that worst case χ_α is as hard as average case χ_β when β is chosen uniformly at random from $\mathbb{Z}_N$. That is, we show an efficient reduction algorithm that given $N, (x_i, f(x_i))_{i=1}^m$ for f a noisy α-character, computes an instance $N, (x'_i, f'(x'_i))_{i=1}^m$ such that f' is a noisy β

character for β distributed uniformly at random in $\mathbb{Z}_N$, and such there is an efficient mapping from β to α.

Moreover, we show a reduction that maps Boolean f to Boolean f', thus also showing that rLCN in Boolean ℓ_2 noise is random self reducible.

Theorem 4.9. *rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ in ℓ_2 noise model is random self reducible. Moreover, it is random self reducible even in the Boolean ℓ_2 noise model.*

Algorithm for rLCN in ℓ_2 Noise

For the ℓ_2 noise model where we conjecture rLCN to be intractable, we present an algorithm for rLCN running in time $N^{1-O(1/(\log\log N)^2)}$. This gives a slight improvement over the $O(N \log N)$ running time of the naive algorithm. (Where the naive algorithm is an algorithm measuring correlation of $O(\log N)$ random samples with each character in G, outputting the character with highest correlation.)

Theorem 4.10 (algorithm for rLCN)**.** *There is an probabilistic algorithm that solves rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ in the Boolean ℓ_2 noise model of parameter $\varepsilon = O(1)$ running in time $N^{1-poly(1/\log\log N))}$.*

This algorithm is based on two ingredients. The first ingredient is our algorithm solving LCN in *interval access model* (see details in chapter 5). In this access model, the algorithm receives samples of the form $\{(x_i, f(x_i))\}_{i=1}^k$ where $x_1, \ldots, x_k$ all fall in a length 2^ℓ interval of $\mathbb{Z}_N$ for a random ℓ in $0, \ldots, \log N$. The second ingredient is an algorithm simulating interval access in the random samples access. This algorithm simply waits until random samples happen to fall in small intervals. In a "birthday sampling lemma" we analyze the number of random samples requires in order to obtain a sufficient number of samples simulating interval access, showing it is $N^{1-O(1/(\log\log N)^2)}$.

4.2.2 rLCN in Noise Models Simulating Product Group Structure

We define noise models simulating product group structure, and survey our results on rLCN in these noise models.

We restrict our attention to functions over groups $\mathbb{Z}_N$ where $N = p_1 p_2 \ldots p_t$ is a product of $t > 1$ distinct primes. In this case, there is an isomorphism[5] between $\mathbb{Z}_N$ and the direct product group $\mathbb{Z}_{p_1} \times \ldots \times \mathbb{Z}_{p_t}$. With this direct product structure in mind, we consider analogies between rLCN and LPN leading to our noise models definitions.

Recall that LPN is defined with respect to functions over the direct product group $\mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2 = \{0,1\}^t$. For each $\bar{\alpha} = (\alpha_1, \ldots, \alpha_t), \bar{x} = (x_1, \ldots, x_t) \in \{0,1\}^t$, recall

[5]For $N = p_1, \ldots, p_t$ a product of $t > 1$ distinct primes, the isomorphism between $\mathbb{Z}_N$ and $\mathbb{Z}_{p_1} \times \ldots \times \mathbb{Z}_{p_t}$ is defined by mapping each $x \in \mathbb{Z}_N$ to $(x_1, \ldots, x_t)$ s.t. $x_i \equiv x \mod p_i$. This transformation as well as its inverse are both efficiently computable, where the inverse transformation in computed using the Chinese Remainder Theorem.

that the $\bar{\alpha}$-character over $\{0,1\}^t$ is defined by

$$\bar{\chi}_{\bar{\alpha}}(\bar{x}) = \prod_{i=1}^{t} \chi_{\alpha_i}(x_i)$$

for $\chi_{\alpha_i}(x_i) = (-1)^{\alpha_i x_i}$ is the α_i-character of the group $\mathbb{Z}_2$. The noisy character $\bar{f}$ of rLPN is a noisy version of $\bar{\chi}_{\bar{\alpha}}$ in Hamming noise model.

Analogously to $\bar{\chi}_{\bar{\alpha}}$, in $G \cong \mathbb{Z}_{p_1} \times \ldots \times \mathbb{Z}_{p_t}$ we define for every $\alpha \in G$ a function

$$\varphi_\alpha(x) = \prod_{i=1}^{k} \chi_{p_i,\alpha_i}(x_i)$$

for χ_{p_i,α_i} the α_i-character of the group $\mathbb{Z}_{p_i}$ and $\alpha_i = \alpha \mod p_i$, $x_i = x \mod p_i$. Analogously to $\bar{f}$ above, we define f to be a noisy version of φ_α (in any bounded distance noise model). We name this noise the *combined primes* noise model. We use the terminology *$\mathcal{S}$-noisy combined primes noise model* for $\mathcal{S}$ some bounded distance noise (say, $\mathcal{S}$ the Hamming noise) to specify that φ_α is corrupted by noise model $\mathcal{S}$.

We also define a related (and seemingly less challenging) noise model, where instead of combining all χ_{p_i,α_i} to one value by taking their product, the algorithm is given random samples access to a noisy version of χ_{p_i,α_i} for a random $i \in 1,\ldots,t$. Namely, the algorithm receives random samples of the form $(x, i, f_i(x))$ for $x \in \mathbb{Z}_N$ and $i \in [t]$ both chosen uniformly at random, and f_i is a noisy version of χ_{p_i,α_i} in one of the bounded distance noise models. We name this noise the *distinct primes* noise model. We use the terminology *$\mathcal{S}$-noisy distinct primes noise model* for $\mathcal{S}$ some bounded distance noise (say, $\mathcal{S}$ the Hamming noise) to specify that the character χ_{p_i,α_i} are corrupted by noise model $\mathcal{S}$.

Definition 4.11 (Noise simulating product group structure)**.** *Let $N = \prod_{i=1}^{t} p_i$ a product of $t > 1$ distinct primes, $f\colon \mathbb{Z}_N \to \mathbb{C}$ a complex function over $\mathbb{Z}_N$, and $\mathcal{S}$ be a noise model from Definition 4.3.*

We use the following notation. For each $\alpha \in \mathbb{Z}_N$, $\alpha_i \equiv \alpha \mod p_i$, let $\chi_{p_i,\alpha_i}\colon \mathbb{Z}_{p_i} \to \mathbb{C}$ is the α_i-character of the group $\mathbb{Z}_{p_i}$, let $\varphi_{N,\alpha}\colon \mathbb{Z}_N \to \mathbb{C}$ be defined by $\varphi_{N,\alpha}(x) = \prod_{i=1}^{k} \chi_{p_i,\alpha_i}(x_i)$ and let $\varphi_{N,\alpha}^{rand}\colon \mathbb{Z}_N \to [t] \times \mathbb{C}$ be defined by $\varphi_{N,\alpha}^{rand}(x) = (i, \chi_{p_i,\alpha_i}(x_i))$ for $i \in [t]$ chosen uniformly at random independently for each $x \in \mathbb{Z}_N$.

We define the following noise models:

- **$\mathcal{S}$-noisy combined primes***: We say that f is a noisy α-character in the $\mathcal{S}$-*noisy combined primes noise model*, if f is a corruption of $\varphi_{N,\alpha}$ in noise model $\mathcal{S}$.*

- **$\mathcal{S}$-noisy distinct primes***: We say that f is a noisy α-character in the $\mathcal{S}$-*noisy distinct primes noise model*, if f is a corruption of $\varphi_{N,\alpha}^{rand}$ in noise model $\mathcal{S}$.*

Tractability of rLCN with Noise Respecting Group Structure

We investigate the computational difficulty of rLCN in noise simulating product group structure. We show that the tractability of rLCN in the combined primes and distinct primes noise models depend on the factorization of N.

Theorem 4.12. *Let $\mathcal{I} = \{N \mid N$ is a product of $t > 1$ distinct primes $p_1, \dots, p_t\}$. Let $\mathcal{S}$ be one of the noise models in Theorem 4.5.*

1. *If N can be efficiently factored, then rLCN over $\{\mathbb{Z}_N\}_{N\in\mathcal{I}}$ in $\mathcal{S}$-noisy distinct primes noise model is in BPP.*
2. *If $\sum_{i=1}^{t} \frac{N}{p_i}$ is co-prime to N and it can be efficiently computed, then rLCN $\{\mathbb{Z}_N\}_{N\in\mathcal{I}}$ in $\mathcal{S}$-noisy combined primes noise model is in BPP.*

Remark 4.13. *The factorization of N or $\sum_{i=1}^{t} \frac{N}{p_i}$ can be given as advice instead of being computed by the algorithm.*

4.3 Proof of Tractability Results

In this section we prove our tractability results. We describe the algorithms solving rLCN over $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ in each of the noise models considered in Theorems 4.5 and 4.12. All algorithms and analyzes are simple, computation details are sometimes only sketched.

In the following we fix $N \in \mathbb{N}$ and $f\colon \mathbb{Z}_N \to \mathbb{C}$ s.t. the input to our algorithms is N and random samples access to f. Denote $\mathbb{T}_N = \mathbb{T} \cap \left\{\omega_N^k\right\}_{k\in\mathbb{N}}$ for $\omega_N = e^{i\frac{2\pi}{N}}$. We assume without loss of generality that $f(x) \in \mathbb{T}_N$, otherwise, we can round $f(x)$ to the closest value in $\mathbb{T}_N$.

Proof of Theorem 4.5

No noise model. Consider first the no noise model. The algorithm for finding α given a random sample $(x, \chi_\alpha(x))$ is as follows: If x is co-prime to N, translate $\chi_\alpha(x)$ to $z = \alpha x \mod N$ (this is by translating the complex number $\chi_\alpha(x)$ to its polar representation of angle θ and radius 1, and computing $\alpha = \theta \cdot \frac{N}{2\pi}$), and output $z \cdot x^{-1}$ mod N, else, fail. The success probability is $1 - \frac{\left|\mathbb{Z}_N \setminus \mathbb{Z}_N^*\right|}{|\mathbb{Z}_N|}$. Repeating the algorithm on fresh samples, the success probability can be amplified.

Hamming noise model of parameter ε. Consider next the Hamming noise model of parameter ε. The algorithm for finding α is as follows. Ask for $m = O(1)$ random samples $(x, f(x))$. For each sample $(x, f(x))$ compute a guess for α using the above algorithm for the no noise case. Output all α appearing in more than $1 - \varepsilon - O(1)$ fraction of the guesses. This algorithm succeed with high probability: let $\alpha \in \mathbb{Z}_N$ be s.t. $f(x) = \chi_\alpha(x)$ on $1 - \varepsilon$ fraction of the $x \in \mathbb{Z}_N$, then $f(x) = \chi_\alpha(x)$ on $1 - \varepsilon - O(1)$ fraction of the samples with probability at least $O(1)$. We remark that in case $\varepsilon > \frac{1}{2}$, this algorithm returns a unique α, whereas if $\varepsilon \le \frac{1}{2}$, the algorithm returns a list of length at most $O(1/\varepsilon)$ containing with high probability all α s.t. $\Delta(\chi_\alpha, f) < \varepsilon$.

This algorithm extends to handle stochastic Hamming noise models with parameter $\varepsilon' < \varepsilon - \rho$ for a non negligible function ρ. This is because for a noisy α-character f in stochastic Hamming noise of parameter ε', $\Delta(f, \chi_\alpha) \le \varepsilon$ with overwhelming probability for Δ the relative Hamming distance.

ℓ_∞ noise model of parameter ε. Consider third the ℓ_∞ noise model with parameter ε. In this noise model, it may be the case that *all samples are noisy* so we cannot use the above algorithm. The algorithm finding for α is as follows. Round the noise parameter ε to the closest value in $\varepsilon' \in \mathbb{T}_N$ and map ε' to the corresponding $M \in \mathbb{Z}_N$ (this is by translating the complex number ε' to its polar representation of angle θ and radius 1, and computing $M = \theta \cdot \frac{N}{2\pi}$). Likewise, for each sample $(x, f(x))$ map $f(x) \in \mathbb{T}_N$ to the corresponding value $f'(x) \in \mathbb{Z}_N$. By definition of the ℓ_∞ noise model, for each sample $(x, f'(x))$, it holds that $\chi_\alpha(x) \in \{f'(x) - M, \ldots, f(x) + M\}$. By constraint on ε, $M \le \mathit{poly} \log N$ and thus we can exhaustively try all values of $(x, f'(x) + s)$ for $s = -M, \ldots, M$ until reaching the correct value $(x, \chi_\alpha(x))$ from which we can find α using the algorithm for the no noise case above. We can verify that a guess α' is correct by checking that $|f'(x) - (a'x \mod N)| \le M$ for sufficiently many samples.

Stochastic ℓ_∞ noise model of parameters ε, σ. Finally consider the stochastic ℓ_∞ noise model of parameters ε, σ. The algorithm for finding α in this noise model is the same as the algorithm for the ℓ_∞ model, only replacing the parameter ε with $\varepsilon' = \varepsilon + O(\sqrt{m})\sigma$, where m is the number of samples requested by the algorithm for the ℓ_∞ noise model of parameter ε. This algorithm succeeds, because for such ε', with high probability, for all m samples $|f(x) - \chi_\alpha(x)| \le \varepsilon$. □

Proof of Theorem 4.12

Let $p_1, \ldots, p_t$ be a factorization of N to $t > 1$ distinct primes.

Distinct primes noise model. Consider the distinct primes noise model, and assume the factorization on N is given as advice to the algorithm. Recall that the samples are of the form $(x, f(x))$ where $f(x) = (i, \chi_{p_i,\alpha_i}(x_i))$, x is uniformly random in $\mathbb{Z}_N$ and i is uniformly random in $1, \ldots, t$. The algorithm for finding α is as follows. Ask for $m = O(\log N)$ samples. For each $i = 1, \ldots, t$, denote by S_i be the set of all samples $(x, (i, \chi_{p_i,\alpha_i}(x_i)))$. For each $i = 1, \ldots, t$, run our algorithm for the no noise model on input p_i and samples S_i; denote its output by α_i. Apply the Chinese Remainder Theorem to combine all these values $\alpha_1, \ldots, \alpha_t$ to $\alpha \mod N$.

To show that the this algorithm succeed, observe first that for $m = O(\log N)$, with high probability, $|S_i|$ is greater than some constant for all $i = 1, \ldots, t$. In this case, with high probability the no noise algorithm succeeds in outputting α_i on each of its applications. In this case, applying the Chinese Remainder Theorem would indeed return α.

Noisy distinct primes noise model. Consider next the $\mathcal{S}'$-noisy distinct primes noise model, where $\mathcal{S}'$ is one of the noise models considered in theorem 4.5. Assume again the factorization of N is given as advice to the algorithm. In these settings an algorithm similar to the above works, only that we replace the ingredient where we ran the algorithm for the no noise model, with an algorithm for $\mathcal{S}'$ noise model. Analysis follows from the success probability of the algorithm for $\mathcal{S}'$, similarly to the no noise case.

Combined primes noise model. Consider the combined primes noise model. Observe that for a divisor p_i of N, $\omega_{p_i} = \omega_N^{N/p_i}$, and thus $\chi_{\alpha,p_i}(x) = (\omega_N^{\alpha x})^{N/p_i}$. Therefore we can rewrite $\varphi_\alpha(x)$ as follows.

$$\varphi_\alpha(x) = \prod_{i=1}^{t} \chi_{\alpha,p_i}(x) = \omega_N^{\alpha x \cdot \sum_{i=1}^{t} \frac{N}{p_i}}$$

Consider the case that $\sum_{i=1}^{t} \frac{N}{p_i}$ is co-prime to N. In this case, if $\sum_{i=1}^{t} \frac{N}{p_i}$ is known to the algorithm, then it operates as follows. Compute $\left(\sum_{i=1}^{t} \frac{N}{p_i}\right)^{-1} \mod N$ and $x^{-1} \mod N$ (using Extended Euclid algorithm), compute $\alpha x \cdot \sum_{i=1}^{t} \frac{N}{p_i}$ (this is by representing the complex value $f(x)$ in a polar representation with radius 1 and angle θ and computing $\theta \cdot \frac{N}{2\pi}$). Output the product of these three values.

This algorithm succeeds provided that there exists a sample $(x, f(x))$ s.t. x is co-prime to N. For a single sample $(x, f(x))$, this happens with probability $1 - \frac{\left|\mathbb{Z}_N \setminus \mathbb{Z}_N^*\right|}{|\mathbb{Z}_N|}$. By taking several sample success probability can be amplified.

Noisy combined primes noise model. Consider next the $\mathcal{S}'$-noisy combined primes noise model, where $\mathcal{S}'$ is one of the noise models considered in theorem 4.5. Assume again that $\sum_{i=1}^{t} \frac{N}{p_i}$ is given as advice to the algorithm. In these settings, we combine algorithms for noise models of Theorem 4.5 with the above algorithm for the combined primes noise model. Specifically, we first apply the algorithm for the $\mathcal{S}'$ noise to find correct values of $(x, \varphi_\alpha(x))$, and then apply our algorithm for the combined primes noise to deduce α from $(x, \varphi_\alpha(x))$. Analysis follows from the analyzes of both algorithms. Let us give two examples.

In the Hamming noise model with parameter ε, we apply the above algorithm for the combined primes noise model on each sample $(x, f(x))$, and output all values appearing in $1 - O(\varepsilon)$ fraction of the samples.

In the ℓ_∞ noise model with parameter ε, for each sample $(x, f(x))$, we map the complex value $f(x)$ to the corresponding element $f'(x) \in \mathbb{Z}_N$. We round ε to $\varepsilon' \in \mathbb{T}_N$ and map ε' to a corresponding element in $M \in \mathbb{Z}_N$. We then apply our algorithm for the combined primes noise model on the set of samples $(x, f'(x) + s)$ for each $s = -M, \ldots, M$.

□

Remark 4.14. *In the distinct primes noise model, we gave an algorithm provided that the factorization of N is known (or can be efficiently found). Even if the factorization of N is hard to find, our algorithm can still return $\alpha \mod p_i$ for all $i = 1, \dots, t$ (despite not knowing $p_1, \dots, p_t$). We do not know however how to combine these values into α without knowing $p_1, \dots, p_t$.*

4.4 Proof of Random Self Reducibility Result

In this section we prove Theorem 4.9 on the random self reducibility of rLCN.

To prove that worst case rLCN is reducible to average case rLCN, we define a transformation mapping samples $(x, f(x))$ for f a noisy α-character to samples $(x', f'(x'))$ for f' a noisy β-character s.t. β is uniformly random from $\mathbb{Z}_N$ and s.t. given β we can efficiently find α. We consider two transformations. Both satisfy the above. One transformation also has the property that if f is Boolean than so if f', namely, it proves random self reducibility within the Boolean ℓ_2 noise model.

The first transformation is as follows. Choose a random $\alpha' \in \mathbb{Z}_N$, and map each $(x, f(x))$ into

$$(x, f(x)) \mapsto (x, f(x)\chi_{\alpha'}(x))$$

This maps f a noisy α character to f' a noisy β character for $\beta = \alpha + \alpha'$ where if f was noisy in ℓ_2 noise model with parameter ε, the so is f'. To see this, first note that $\chi_\alpha \chi_{\alpha'} = \chi_{\alpha+\alpha'}$. Second, note that for $\eta = f - \chi_\alpha$, $\eta' = f' - \chi_{\alpha'}$ denoting the noises patterns of f and f', respectively, $\eta' = \eta\chi_{\alpha'}$, which implies that $|\eta'(x)| = |\eta(x)|$ $\forall x \in \mathbb{Z}_N$, and therefore, $\|\eta'\|_2 = \|\eta\|_2$.

We remark that this transformation also preserves distance in the no noise, Hamming noise, ℓ_∞ noise and stochastic ℓ_∞ noise models.

The second transformation we consider is as follows. Choose a random $\alpha' \in \mathbb{Z}_N^*$, and map each $(x, f(x))$ into

$$(x, f(x)) \mapsto (x\alpha', f(x))$$

The effect of this transformation on the Fourier coefficients is that $\widehat{f'}(\alpha) = \widehat{f}(\alpha/\alpha')$. Therefore, if f was a noisy α character, then f' is a noisy β character for $\beta = \alpha/\alpha'$, and the distance remains unchanged, be it with respect to Hamming, ℓ_∞ or ℓ_2 measure. The samples locations $x_1\beta^{-1}, \dots, x_m\beta^{-1}$ are independent random locations in $\mathbb{Z}_N$ if so were $x_1, \dots, x_m$.

The advantage of the second transformation over the first is that $\{f'(x)\}_{x\in\mathbb{Z}_N} = \{f(x)\}_{x\in\mathbb{Z}_N}$. In particular, if f is Boolean than so is f'. Namely, this transformation gives random self reducibility within the restricted class of Boolean functions f. □

4.5 Algorithm for rLCN: Proof of Theorem 4.10

In this section we prove Theorem 4.10 giving an algorithm solving rLCN in time $N^{1-O(1/(\log\log N)^2)}$ in the ℓ_2 noise model of parameter $\varepsilon = O(1)$.

Recall that the SFT algorithm executes $\log N$ refinement, and in each step $\ell = 1, \ldots, \log N$, it requires $k = O((\log\log N)^2)$ samples chosen uniformly at random from an interval of length 2^ℓ. According to the birthday sampling lemma below, we can simulate the samples needed for all steps ℓ using

$$\sum_{\ell=1}^{\log N-1} k \cdot \left(\frac{N}{2^\ell}\right)^{1-\frac{1}{k}} = O(N^{1-\frac{1}{k}} \log N)$$

random samples. Namely, with running time $O(N^{1-O(1/(\log\log N)^2)} \log N)$ we can run the SFT algorithm in the random samples access model, obtaining an algorithm for rLCN.

Lemma 4.15 (Birthday sampling lemma)**.** *For any $\delta > 0$, $N, k \in \mathbb{N}$ and $\ell \in \{0, \ldots, \log N\}$, a random subset $X \subseteq \mathbb{Z}_N$ of size $|X| > k \cdot \left(\delta \cdot \frac{N}{2^\ell}\right)^{1-\frac{1}{k}}$ contains k elements all falling in a length 2^ℓ interval of $\mathbb{Z}_N$, with probability at least δ.*

Proof. Denote $m = |X|$. The probability that X contains k samples at a fixed interval of length 2^ℓ is at least $C(m,k)\left(\frac{2^\ell}{N}\right)^k$. Partitioning $\mathbb{Z}_N$ to $\frac{N}{2^\ell}$ *disjoint* length 2^ℓ intervals, the probability that X contains k samples in one of those intervals is at least $\frac{N}{2^\ell} \cdot C(m,k) \cdot \left(\frac{2^\ell}{N}\right)^k \geq \left(\frac{m}{k}\right)^k \cdot \left(\frac{2^\ell}{N}\right)^{k-1}$. This is greater than δ iff $m > k \cdot \left(\delta \cdot \left(\frac{N}{2^\ell}\right)^{k-1}\right)^{1/k}$.
□ □

Chapter 5

Learning Characters with Noise in Intermediate Access Models

In this chapter we explore the problem of *Learning Characters with Noise* in access models which are not as permissive as the query access model and yet not as restrictive as the random samples access model.

The input to the problem of learning characters with noise (LCN) are samples $\{(x_i, f(x_i))\}_{i=1}^m$ of a function $f\colon G \to \mathbb{C}$ over a group G and the output is the significant Fourier coefficients of f, that is, all $\alpha \in G$ s.t. $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$. The *access model* governs the mechanism by which the samples $\{(x_i, f(x_i))\}_{i=1}^m$ are given to the learning algorithm.

In chapters 3 and 4 we considered the *query access model* and the *random samples access model*. In this chapter we consider *intermediate access models*, in which the x_i's are neither at the complete discretion of the learning algorithm (as was the case for the query access model), nor completely random (as was the case for the random samples access model). Access models that we consider include: *interval access* where the samples are correlated by having all x_i's drawn from the same (randomly chosen) interval, *GP-access* where $x_1, \ldots, x_m$ form a geometric progression in the group G, and *subset access* where the $x_1, \ldots, x_m$ are restricted to lie in a predetermined subset $Q \subseteq G$.

We explore the complexity of LCN in the above intermediate access models and present applications of our findings. In the interval access model, we show that LCN is tractable. An application of this result is a (non-polynomial time) algorithm for solving LCN in random samples access model discussed in chapter 4. In the subset access model, we show that for any family $\mathcal{F}$ of functions there are subsets Q of size polynomial in $\log|G|, \log|\mathcal{F}|$ such that LCN with subset access to Q is tractable for any function f in $\mathcal{F}$. Applications of this tractability results are decoding algorithms for polynomial rate codes, discussed in chapter 6. In the GP-access model, we show that tractability of LCN is related to the problem of proving bit security of the Diffie-Hellman (DH) function. In particular, we show that either LCN in GP-access model is intractable, or several bits of DH are as secure as DH. This places use in a "win-win" situation from the perspective of finding hard problems for cryptographic protocols.

5.1 Introduction

In this chapter we explore the complexity of the Learning Characters with Noise problem in access models which are not as permissive as the query access model and yet not as restrictive as the random samples access model.

The input to the problem of learning characters with noise (LCN) is a description of a finite abelian group G, a threshold $\tau \in \mathbb{R}^+$ and samples $\{(x_i, f(x_i))\}_{i=1}^m$ of a complex function $f\colon G \to \mathbb{C}$ over the group G. The output is the τ-significant Fourier coefficients of f, that is, all α s.t. the α Fourier coefficient has weight $\left|\widehat{f}(\alpha)\right|^2$ at least τ. The *access model* governs the mechanism by which the samples $\{(x_i, f(x_i))\}_{i=1}^m$ are given to the learning algorithm. The access models we consider are not as permissive as the query access model, because the x_i's are not at the discretion of the learning algorithm. The are also not as restrictive as the random samples access model, because the x_i's are not completely random.

Definition 5.1 (LCN in access model M)**.** *The input to LCN in M-access model is a description of a finite abelian group G by its generators and their orders, a threshold $\tau \in \mathbb{R}^+$ and M-access to a function $f\colon G \to \mathbb{C}$, that is, samples of f are generated via oracle access to algorithm M; the goal is to find all τ-significant Fourier coefficients of f.*[1]

The motivation for considering intermediate access models arises from applications where it is impossible to gain query access to the function whose significant Fourier coefficients are of interest, and yet, it is possible to obtain samples that are not merely random.

Access models we consider include the following examples. We consider the *GP-access* model, where samples $\{(x_i, f(x_i))\}_{i=1}^m$ satisfy that $x_1, \ldots, x_m$ form a geometric progression in the group G. We relate the tractability of LCN in GP-access model to bit security of the Diffie-Hellman (DH) function, showing that either LCN in GP-access model is intractable, or several bits of DH are as secure as DH (details in chapter 7).

We consider the *subset access* model, where samples $\{(x_i, f(x_i))\}_{i=1}^m$ satisfy that $x_1, \ldots, x_m$ are restricted to lie in a predetermined subset $Q \subseteq G$, and are otherwise completely in the power of the learning algorithm. We show that there exists polynomial size sets Q s.t. LCN with subset access to Q is tractable provided that the input functions is drawn from a predetermined family of functions. This algorithm is a component in an efficient decoding algorithms in Binary Symmetric Channel noise model we give for polynomial rate codes discussed in chapter 6.

For functions over $\mathbb{Z}_N$,[2] we consider *interval access* model. In this access model, the samples $\{(x_i, f(x_i))\}_{i=1}^m$ are correlated by having all x_i's drawn from the same (randomly chosen) interval. We show that LCN in interval access model is tractable. This algorithm is a component in our (non-polynomial time) algorithm for solving LCN in random samples access model (details in chapter 4).

[1]We say that $\alpha \in G$ is a *τ-significant* Fourier coefficient of f iff $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$.

[2]$\mathbb{Z}_N$ denotes the additive group of integers modulo N

Related Works

For the special case case when the functions f are over the Boolean Cube $\{0,1\}^n$, intermediate access models were previously considered for the problem of Learning Parity with Noise (LPN). One example of an intermediate access model studied for LPN is the *random walk model* where samples $\bar{q} \in \{0,1\}^n$ are generated by a random walk on the hypercube (see [21] and references therein).

Another example where intermediate access models were (implicitly) studied for LPN is the subset access model. Specifically, we observe that every linear error correcting code defines a subset $Q \subseteq \{0,1\}^n$ such LPN with query access to Q is tractable when restricted to functions f that are the corrupted codewords. Let us elaborate. A linear code is defined by a generating matrix $A \in \{0,1\}^{k \times n}$ such that encoding of messages $x \in \{0,1\}^k$ is by codewords $xA \in \{0,1\}^n$. Let $Q \subseteq \{0,1\}^k$ be the set of all columns of A. We observe that the decoding algorithm solves Q-LCN on the restricted class of functions whose restriction to Q is $xA + \eta$ for some $x \in \{0,1\}^k$ and for η a noise vector. Since there are linear codes of constant rate,[3] then there are subsets $Q \subseteq \{0,1\}^k$ of linear size $|Q| = O(k)$ such that LPN with subset access to Q is tractable (with restricted input functions).

By the above, when restricting attention to functions over the Boolean cube $\{0,1\}^n$, there are linear size subsets $Q \subseteq \{0,1\}^k$ such that LCN with subset access to Q is tractable (with restricted input functions). For functions over arbitrary finite abelian groups G, we show that there are subsets $Q \subseteq G$ of polynomial size $|Q| = \mathit{poly} \log |G|$ s.t. LCN is tractable with subset access to Q (again, with restricted input functions). Finding sets Q of linear size $|Q| = O(\log |G|)$ is an open problem. The problem is open even for the case $G = \mathbb{Z}_N$.

5.2 Overview of Definitions and Results

We consider the access models of *interval access*, *GP-access*, *subset access* and *DH-access*, defined below, and explore the complexity of LCN in these access models. We show that LCN is tractable in interval access model; relate the tractability of LCN in GP-access and DH-access models to bit security of the Diffie-Hellamn function; and show that there exists polynomial size sets Q s.t. LCN with subset access to Q is tractable when the input functions is drawn from a restricted family of functions.

The considered access models are intermediate access models in the sense that they are strictly less powerful than the query access model and strictly more powerful than the random samples access model.

Proposition 5.2. *The interval access and GP-access models are strictly stronger than the random samples access model, and are strictly weaker than the query access model. The subset access model to subset $Q \neq \phi, G$ is incomparable to the random samples access model (i.e., it is neither stronger nor weaker), and is strictly weaker than the query access model.*

[3]A linear error correcting code has *constant rate* if its generating matrix has dimension $k \times n$ for $n = O(k)$

Proof. Proof is deferred to section 5.3 □

In the following we formally define these access models, discuss our results regarding the complexity of LCN in these access models, and mention applications of our results.

5.2.1 Interval Access Model

In the *interval access* model, the samples $\{(x_i, f(x_i))\}_{i=1}^m$ are correlated by having all x_i's drawn from the same (randomly chosen) interval.

Definition 5.3 (Interval access). *Let $f\colon \mathbb{Z}_N \to \mathbb{C}$ be a complex function over the additive group $\mathbb{Z}_N$ of integers modulo N.* Interval access *to f is given by an algorithm M that, on input m, outputs samples*

$$((x, \ell, (y_1, \ldots, y_m)), f(x + y_1), \ldots, f(x + y_m))$$

for $x \in \mathbb{Z}_N$ chosen uniformly at random, $\ell \in \{1, \ldots, \log N\}$ chosen uniformly at random, and $y_1, \ldots, y_m$ in the interval $\left\{0, \ldots, 2^\ell - 1\right\}$ chosen independently and uniformly at random; and its running time is polynomial in m and $\log N$.

We show that LCN is tractable in the interval access model, namely, there is an algorithm solving LCN in this access model in time polynomial in $\log N$ and $1/\tau$.

Theorem 5.4. *LCN in interval access model is in BPP.*

Proof. Examining the SFT algorithm from Chapter 3 reveals that it did not use the full power of the query access model. The queries it makes are in fact chosen independently at random from random intervals of size 2^ℓ for $\ell = 1, \ldots, \log N$. Such queries can be simulated using *poly*$(\log N, 1/\tau)$ samples generated in the interval access model. □

Remark 5.5 (random walk interval access). *For functions $f\colon G \to \mathbb{C}$ over a product of cyclic groups $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, the interval access model can be extended to a* random walk interval (RWI) access model, *and LCN is tractable in this RWI-access model. The RWI-access model combines the interval access model with the random walk access model as follows. We think of G as a k dimensional cube with axis i of length N_i, and samples are given via a random walk on the nodes of the cube where in each node $(x_1, \ldots, x_k)$, some axis $i \in [k]$ is chosen uniformly at random, and m samples $((y_1, \ldots, y_k), f(y_1, \ldots, y_k))$ are given with $y_j = x_j$ for all $j \neq i$ whereas y_i are values chosen according to the interval access model in $\mathbb{Z}_{N_i}$.*

LCN in the RWI-access model is tractable by an algorithm combining our algorithm for LCN in interval access model together with a generalization of the algorithm Bshouty et.al. [21] gave for function over the Boolean cube $\{0,1\}^n$.

An analogous access model and a corresponding learning algorithm can be given for any finite abelian group, when given its generator and their orders. This is by relying on the isomorphism between finite abelian groups and product of cyclic groups.

5.2.2 GP-Access Model and DH-Access Model

In the *GP-access* model the samples $\{(x_i, f(x_i))\}_{i=1}^m$ satisfy that $x_1, \ldots, x_m$ form a geometric progression in the group G.

Definition 5.6 (GP-Access). *Let $f\colon G \to \mathbb{C}$ be a complex function over an abelian group G.* GP-access to f *is given by an algorithm M that, on input integers $k_1, k_2, ..., k_m$, outputs samples*

$$(r, \{f(r^{k_i})\}_{i=1}^m)$$

for $r \in G$ distributed uniformly at random and where the power operation is computed in the group G; and its running times is polynomial in m and $\log|G|$.

Another access model we consider is the *DH-access* model. The *DH-access* model arises in our study of cryptographic hardcore predicates for the Diffie-Hellman function,[4] and its samples are tailored for that application.

Definition 5.7 (DH-access). *Let $f\colon \mathbb{Z}_N \to \mathbb{C}$ be a complex function over the additive group $\mathbb{Z}_N$ of integers modulo N.* DH-access to f *is given by an algorithm M that, on input a prime $p \in \mathbb{N}$, a generator g of $\mathbb{Z}_p^*$ and $g^a, g^b, g^{a'}, g^{b'} \in \mathbb{Z}_p^*$, outputs $f(g^{a'b'}/g^{ab})$ in time T, for T the time of computing $g^{a'}, g^{b'}$ given p, g, g^a, g^b.*

Determining whether LCN in GP-access or DH-access is tractable is an open problem. Using known (non polynomial time) algorithms for computing the Diffie-Hellman function, it is possible to reduce (in non-polynomial time) LCN in DH access model to LCN in query access model (qLCN). Combined with our SFT algorithm solving qLCN in time polynomial in $\log N$ and $1/\tau$ this gives an algorithm solving LCN in DH-access in times polynomially related to the time of computing the Diffie-Hellman function, currently bounded by $2^{O\left((logN)^{1/3}(\log\log N)^{2/3}\right)}$.

Theorem 5.8. *There is an algorithm solving LCN in DH-access model in time $T_{DH} \cdot poly(\log N, 1/\tau) \leq 2^{O\left((\log N)^{1/3}(\log\log N)^{1/3}\right)}$ for T_{DH} the time for computing the DH function.*

Proof. To solve LCN in DH-access model we first reduce this problem to LCN in query access model in time T_{DH}, and then apply the SFT Algorithm 3.4 to solve LCN in the query access model in time $poly(\log N, 1/\tau)$. Combined together, this gives the running time stated in the theorem.

Reducing LCN in DH-access model to LCN in query access model, is by using DH-access oracle M to simulate query access oracle M' as follows. For any $R \in \mathbb{Z}_p^*$, we define

$$M'(p, g, g^a, g^b, R) = M(DH(g^a, g^b) \cdot g^r, g)$$

Observe that $DH(g^a, g^b) \cdot R = g^{ab+r}$ for r the Discrete Logarithm of R, that is, $g^r = R$. By the definition of the DH-access model M, this implies that that the output of $M' = M(g^{ab+r})$ is the value of f on input R, because $g^{(ab+r)\cdot 1 - ab} = R$.

[4]The Diffie-Hellman (DH) function is the function $DH_{p,g}\colon \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ for p a prime and g a generator of $\mathbb{Z}_p^*$, defined by $DH_{p,g}(g^a, g^b) = g^{ab}$ for any $g^a, g^b \in \mathbb{Z}_p^*$.

The running time of M' is dominated by the time for T_{DH} computing $DH(g^a, g^b)$. □

We show that tractability of LCN in GP-access or DH-access models would have applications in cryptography. Specifically, it would allow proving security of bits of the Diffie-Hellman function – a long standing open problem (see discussion in chapter 7). This result places us in a "win-win" situation from the perspective of designing cryptographic protocols: Either LCN is easy in GP-access or DH-access model, which would imply a security result regarding the Diffie-Hellman function. Or alternatively, LCN is intractable in these access models, in which case, LCN with these access models poses a new candidate hard function to base cryptographic protocols on.

Theorem 5.9. *If there is a polynomial time algorithm solving LCN in DH-access model or in GP-access model, then every segment predicate (as defined in Definition 7.8) is hardcore for the Diffie-Hellman function.*

Proof. Details are given in Chapter 7 section 7.5. □

5.2.3 Subset Access Model

In the *subset access* model for some fixed and predetermined set $Q \subseteq G$, the learning algorithm is allowed to query functions $f\colon G \to \mathbb{C}$ on any $x \in Q$. We use the terminology *Q-access* referring to subset access when Q is the set from which queries are given.

Definition 5.10 (Q-access)**.** *Let $f\colon G \to \mathbb{C}$ be a complex function over an abelian group G, and $Q \subseteq G$ a subset of G. Q-*access *to f is given by an algorithm M that, on input x in the subset Q, outputs the value $f(x)$ in unit time.*

We focus on a variant of LCN –named, $(Q, \mathcal{F})$-LCN– where the input function is restricted to come from some family $\mathcal{F}$ of complex valued functions over a group G and the queries are taken from a subset $Q \subseteq G$. Asymptotically, this problem is defined with respect to a family of subsets $Q_{N,\tau}$ corresponding to groups G_N of growing sizes and to varying thresholds τ.

Definition 5.11 ($(\mathcal{Q}, \mathcal{F})$-LCN)**.** *Let $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathcal{I}}$ be a family of functions $\mathcal{F}_N \subseteq \{f\colon G_N \to \mathbb{C}\}$ over finite abelian groups G_N, and let $\mathcal{Q} = \{Q_{N,\tau}\}_{N\in\mathcal{I},\tau\in\mathbb{R}^+}$ a family of subsets $Q_{N,\tau} \subseteq G_N$. The input to $(\mathcal{Q}, \mathcal{F})$-LCN is a description of a finite abelian group G_N,[5] $\tau \in \mathbb{R}^+$ and $Q_{N,\tau}$-access to a function $f \in \mathcal{F}_N$; the goal is to find all the τ-significant Fourier coefficients of f.*[6]

We explore the complexity of LCN in Q-access model (Q-LCN, in short). The tractability of this problem depends on the choice of the subset Q. For example, when $Q = G$ is the entire domain, then Q-access is identical to the query access, and Q-LCN is tractable as shown in chapter 3. In contrast, for empty set $Q = \phi$, Q-access

[5]A description of a group G is given by its generators and their orders

[6]We say that $\alpha \in G_N$ is a τ-significant Fourier coefficients of $f\colon G_N \to \mathbb{C}$ if the α Fourier coefficient of f is of weight at least $\left|\widehat{f}(\alpha)\right|^2 \geq \tau$.

provides no information on the function f and thus solving Q-LCN is information theoretically infeasible.

The challenge is to devise small sets Q s.t. Q-LCN is tractable. This goal is motivated by applications where fixed queries have much smaller cost than varying queries, and by applications in coding theory, examples follow.

One motivation for the subset access model is drawn from settings where querying fixed locations is less costly than querying varying locations. As an example, consider functions f measuring existence of seismic activity in various geographic locations. When identifying geographic locations with the group $G = \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$ representing longitudes and latitudes (up to some finite precision), f is a Boolean function over G. It is possible to place in various geographic locations $Q \subseteq G$ sensors/transmitters that would sense whether there is a seismic activity and transmit the findings to a control center. To allow initial placement of sensor/transmitters, the set Q must be of reasonable size. Once sensors in locations Q are fixed, there is only a minor cost in collecting the data $f(q)$ for $q \in Q$. In contrast, it is very expensive to query f on new locations $q \notin Q$, as this requires placing a new sensor in location q. When interested in Fourier analysis of the seismic data f, the challenge in to find small sets Q s.t. Q-LCN is tractable.

Another motivation for the subset access model is taken from coding theory. In chapter 6 we present a new family of error correcting codes –Multiplication Codes– that encode messages $x \in G$ by the string $(P_x(q))_{q\in Q}$ for $\{P_x\}_{x\in G}$ Boolean functions and $Q \subseteq G$. A central component in our decoding algorithm for these codes is an algorithm solving Q-LCN applied on codewords corrupted by noise. When taking $Q = G$, these codes are efficiently decodable due to the tractability of LCN in the query access model, but their encoding uses a great deal of redundancy, namely, it has a bad rate. Taking smaller sets Q, we can achieve much better rate, *e.g.*, for $Q = O(\log|G|)$, the rate is constant. The challenge is to devise small sets Q s.t. Q-LCN is tractable, which would allow efficiently decoding these codes.

We show there exists polynomial size sets Q with which $(Q, \mathcal{F})$-LCN is tractable. That is, for every family of functions $\mathcal{F}$ over a group G, we present polynomial size sets Q s.t. Q-LCN is tractable w.r. to input functions $f \in \mathcal{F}$, and the size of Q is polynomial in $\log|G|$, $\log|\mathcal{F}|$ and $\frac{1}{\tau}$. Moreover, we present a randomized algorithm that efficiently finds such sets Q with high probability.

Our idea for finding polynomial size subsets with which we can solve LCN in the subset access model is as follows. We define Q to be the set of queries asked by our SFT Algorithm (see chapter 3) on a fixed function $f \in \mathcal{F}$, and hope that these queries allow solving LCN on any $f \in \mathcal{F}$, that is, that $(Q, \mathcal{F})$-LCN is tractable. The set Q is indeed of polynomial size, because the SFT algorithm makes only $poly(|G|, 1/\tau)$ queries. For the fixed function f, indeed it is possible to solve Q-LCN, because the SFT algorithm solves LCN when querying f only on $x \in Q$. For other functions $f \in \mathcal{F}$, however, it is not immediately clear whether queries in Q suffice for solving LCN. In particular, our analysis of the SFT algorithm in chapter 3 relies on choosing *fresh queries* for every given function f, whereas, in the subset access model the set of queries Q must be good for *all* functions $f \in \mathcal{F}$. Nevertheless, with a moderate increase in the size Q (essentially, taking $\log(|G_N| \cdot |\mathcal{F}_N|)$ rather than $\log\log|G_N|$

queries), we can show that Q is good *for all* $f \in \mathcal{F}$ with high probability. This is simply by using union bound.

We use the following terminology. Let G_N be a finite abelian group, and $\tau \in \mathbb{R}^+$ a threshold. For $Q_{N,\tau}$ a subset of G_N, we say that $Q_{N,\tau}$ is of *polynomial size* if its size is upper bounded by a polynomial in $\log|G_N|$, $\log|\mathcal{F}_N|$ and $1/\tau$. Let $\mathcal{F}_N \subseteq \{f : G_N \to \mathbb{C}\}$ be a family of functions over G_N, we say that $(Q_{N,\tau}, \mathcal{F}_N)$*-LCN is in P*, if there is an algorithm solving $(Q_{N,\tau}, \mathcal{F}_N)$-LCN for all $f \in \mathcal{F}_N$ and $\tau' \geq \tau$ and its running time is polynomial in $\log|G_N|$, $\log|\mathcal{F}_N|$ and $1/\tau$. Asymptotically, for $\mathcal{F} = \{\mathcal{F}_N\}_N$ and $\mathcal{Q} = \{Q_{N,\tau}\}_{N,\tau}$, we say that $(\mathcal{Q}, \mathcal{F})$*-LCN is in P*, if $(Q_{N,\tau}, \mathcal{F}_N)$-LCN is in P for every N, τ.

Theorem 5.12. *For every family $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathcal{I}}$ of functions $\mathcal{F}_N \subseteq \{f : G_N \to \mathbb{C}\}$ over finite abelian groups G_N, there exists a family of polynomial size subsets $\mathcal{Q} = \{Q_{N,\tau} \subseteq G_N\}_{N\in\mathcal{I},\tau\in\mathbb{R}^+}$ s.t. $(\mathcal{Q}, \mathcal{F})$-LCN is in P.*

Moreover, there is a randomized algorithm that, given a description of the group G_N, a threshold τ and the size $|\mathcal{F}_N|$, outputs a polynomial size subset $Q_{N,\tau} \subseteq G_N$ s.t. with high probability $(Q_{N,\tau}, \mathcal{F}_N)$-LCN is in P.

Proof. Given, a description of the group G_N, $|\mathcal{F}_N|$ and τ, we define Q to be the output of the Generate Queries Algorithm 3.5 on input a description of the group G_N, τ and confidence parameter $\delta = \frac{1}{3|\mathcal{F}_N|}$. We show that the Fixed Queries SFT Algorithm 3.6 is a polynomial time algorithm solving $(Q, \mathcal{F}_N)$-LCN (with high probability over the choice of Q): For any *fixed* function f over G_N, the Fixed Queries SFT Algorithm 3.6 outputs all τ-significant Fourier coefficients of f with probability at least $1 - \delta$ over the choice of Q (see Lemma 3.17). By union bound, this holds for all $f \in \mathcal{F}_N$ with probability at least $1 - |\mathcal{F}_N|\,\delta = 2/3$ □

5.2.4 Further Results

In the applications we consider for LCN in subset access model, it is often natural to consider settings where some of the functions values $f(q)$ are faulty. For example, in the seismic activity example above, it is reasonable to expect some of the sensors to break down with time and stop sending data or send false data. In the coding application the codewords are expected to be corrupted by noise during transmission.

We extend our results to address "faulty" functions, showing there are (efficiently computable) polynomial size sets $\mathcal{Q}$ s.t. $(Q, \mathcal{F})$-LCN is in P even when given a Q-access to a *faulty version* f' of a function $f \in \mathcal{F}$.

We define two models measuring how evenly the faulty values are distributed in the accessed subset Q: "well spread faults" and "δ-spread faults". These models are related to well known noise models: Noise generated by a Binary Symmetric Channel yields (with high probability) well spread faults. Noise generated by an adversary flipping at most δ fraction of the values yields $O(\delta)$-spread faults.

For well spread faults, we show that there are polynomial size sets Q such that faulty $(Q, \mathcal{F})$-LCN is in P. That is, there is an algorithm finding all τ-significant Fourier coefficients of a function $f \in \mathcal{F}$ in time polynomial in $\log|G|, \log|\mathcal{F}|$ and

$1/\tau$, when given Q-access to a functions $f'\colon G \to \{\pm 1\}$ which is a corruption of f by well spread faults.

For δ-spread faults, we show there are polynomial size sets Q such that faulty $(Q, \mathcal{F})$-LCN is solvable in time $|G|^{O(\delta)}$. That is, there is an algorithm finding all τ-significant Fourier coefficients of a function $f \in \mathcal{F}$ in time polynomial in $|G|^{O(\delta)} \cdot$ *poly*$(\log |G|, \log |\mathcal{F}|, 1/\tau)$, when given Q-access to a function $f'\colon G \to \{\pm 1\}$ which is a corruption of f by δ-spread faults.

To simplify the presentation we focus on the case of Boolean functions over $\mathbb{Z}_N$. Results extend to non-Boolean functions over arbitrary finite abelian groups.

In the following $\mathcal{F} = \{\mathcal{F}_N\}_{N\in\mathbb{N}}$ for $\mathcal{F}_N \subseteq \{f\colon \mathbb{Z}_N \to \{\pm 1\}\}$ a family of Boolean functions over $\mathbb{Z}_N$, $\mathcal{Q} = \{Q_{N,\tau} \subseteq \mathbb{Z}_N\}_{N\in\mathbb{N},\tau\in(0,1)}$ is a family of subsets, and $\mathcal{F}_\delta$ is a corruption of $\mathcal{F}$ by δ-spread faults, defined as follows.

Definition 5.13 (δ-spread faults)**.** *Let $Q_{N,\tau} = A - \bigcup_{\ell\in[\log N]} B_\ell$ for $A, B_1, \dots, B_\ell \subseteq \mathbb{Z}_N$. Let $f, f'\colon \mathbb{Z}_N \to \{\pm 1\}$ be Boolean functions over $\mathbb{Z}_N$. We say that f'* is a corruption of f by δ-spread faults *if for at least $(1-\delta)$ fraction of the $\ell \in [\log N]$, $f'(q) = f(q)$ on at least $(1 - O(1))$-fraction of the $q \in A - B_\ell$. That is,*

$$\left|\left\{ \ell \in [\log N] \mid \frac{\#\{q \in A - B_\ell \mid f'(q) \neq f(q)\}}{|A - B_\ell|} > \varepsilon \right\}\right| < \delta \log N$$

for some $\varepsilon = O(1)$ sufficiently small. When $\delta = 0$, we say that the faults are well spread.

Definition 5.14 (δ-Faulty $(\mathcal{Q}, \mathcal{F})$-LCN)**.** *The input to* δ-faulty $(\mathcal{Q}, \mathcal{F})$-LCN *is a description of a group $\mathbb{Z}_N$, a threshold $\tau \in (0,1)$ and Q-access to a function $f'\colon \mathbb{Z}_N \to \{\pm 1\}$ which is a corruption of some function $f \in \mathcal{F}$ by δ-spread faults; the goal is to find all τ-significant Fourier coefficients of f.*

Theorem 5.15 (δ-Faulty $(\mathcal{Q}, \mathcal{F})$-LCN)**.** *For every family $\mathcal{F}$ of Boolean functions over groups $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$, there exists a family of subsets $\mathcal{Q} = \{Q_{N,\tau} \subseteq \mathbb{Z}_N\}_{N\in\mathbb{N},\tau\in(0,1)}$ s.t.*

1. *For $\delta = 0$, δ-Faulty $(\mathcal{Q}, \mathcal{F})$-LCN is in P. That is, there is an algorithm solving δ-faulty $(\mathcal{Q}, \mathcal{F})$-LCN is in running time poly$(\log N, \log |\mathcal{F}_N|, 1/\tau)$.*

2. *For $\delta > 0$, there is an algorithm solving δ-faulty $(\mathcal{Q}, \mathcal{F})$-LCN is in running time $N^{2\delta} \cdot$ poly$(\log N, \log |\mathcal{F}_N|, 1/\tau)$.*

Proof. Proof omitted. ☐

5.3 Omitted Proofs

Proof of Proposition 5.2

We say that access model M is *stronger* than access model M', if samples generated by model M' can be generated in polynomial time in model M. We say that M is *strictly stronger* than M', if M is stronger than M' but M' is not stronger than M. Conversely, we say that access model M is *weaker* than access model M', if M' is (strictly) stronger than M.

Proposition 5.2. The interval access and GP-access models are strictly stronger than the random samples access model, and are strictly weaker than the query access model. The subset access model to subset $Q \neq \phi, G$ is incomparable to the random samples access model (*i.e.*, it is neither stronger nor weaker), and is strictly weaker than the query access model.

Proof. We compare the interval access model to the random samples access model and the query access model. We first argue that the interval access model is strictly stronger than the random samples access model. To see this, first observe that intervals access model is stronger than the random samples model, because random samples can be generated by the interval access model when setting the number of samples to be $m = 1$. Next observe that random samples model is not stronger than intervals access model, because for example, in interval access, when setting $m = 2$, there is probability $1/(\log N + 1)$ of receiving values $f(x)$ and $f(x+1)$ for some $x \in \mathbb{Z}_N$, whereas in contrast in the random samples access, the probability of this event is $1/\sqrt{N}$.

We next argue that the interval access model is strictly weaker than the query access model. For example, in interval access, for each fixed x, the probability of receiving $f(x)$ in time $O(1)$ is $O(1/N)$, whereas in the query access this value can be asked and received in unit time.

We compare the GP-access model to the random samples access model and the query access model. We first argue that GP-access is strictly stronger than the random samples model. GP-access is stronger than random samples access, because for setting $t, k = 1$ it give a sample $(x, f(x))$ for x distributed uniformly at random. Random samples access is not stronger than GP-access, because in the random samples access model the probability of receiving three samples $(x, f(x))$, $(y, f(y))$, $(z, f(z))$ with x, y, z forming a geometric progression is $Pr[y/x = z/x] = 1/N$, whereas such samples are obtained with probability 1 in the GP-access model.

We next argue that GP-access is strictly weaker than query access model. For example, in GP-access model, for each fixed x, the probability of receiving $f(x)$ in time $O(1)$ is $O(1/|G|)$, whereas in the query access this value can be asked and received in unit time.

We compare the Q-access model for a non empty subset Q strictly contained in G to the random samples access model and the query access model. We first show that Q-access is not stronger than random samples access, because for any x not in Q, the probability of getting sample $(x, f(x))$ is zero in Q-access whereas its greater than zero in random samples access model. We next show that random samples access model is not stronger than Q-access model, because for $x \in Q$ the probability of getting sample $(x, f(x))$ in the random samples access model is $1/|G|$, whereas it can be received in unit time in the Q-access model. Finally, Q-access is clearly strictly weaker than query access. □

Chapter 6

Codes for Finite Abelian Groups: Multiplication Codes

In this chapter we present the new class of error correcting codes: *Multiplication codes (MPC)*. We present our decoding algorithms for MPC codes, showing they achieve desirable combinatorial and algorithmic properties, including: (1) binary MPC of constant distance and exponential encoding length for which we provide efficient *local list decoding* and *local self correcting* algorithms; (2) binary MPC of constant distance and polynomial encoding length for which we provide efficient *decoding* algorithm in random noise models; (3) binary MPC of *constant rate and distance* for which we provide (non polynomial time) *decoding* algorithm in adversarial noise model. MPC codes are unique in particular in achieving properties as above while having a large group as their underlying algebraic structure.

6.1 Introduction

Error correcting codes encode messages into codewords in a way that allows decoding, *i.e.*, recovery of the original messages even from codewords corrupted by some noise. Error correcting codes were introduced by Shannon [76] and Hamming [50] for application such as communication over noisy channels or storage over noisy devices, and have since found many applications in other areas such as complexity and cryptography.

The performance of error correcting codes is measured by an array of parameters, where the parameters to be emphasized and optimized vary according to the application in mind. Examples of performance goals arising in various applications follow.

- **Efficiency, rate and distance.** In communication applications, classical goals are to have efficient encoding and decoding algorithms satisfying that the encoding has high rate (*i.e.*, it introduces only little redundancy) and the decoding algorithm handles a wide range of noise patterns. Explicit codes constructions achieving good performance in terms of the above parameters followed the works of [76] and [50], examples of such codes include: Reed-Solomon [72] and Reed-Muller [68,71] codes, Low-Density-Parity-Check codes [34], and expander based codes [7,44,78,80].

- **Local list decoding.** Applications in complexity and cryptography sometimes introduce alternative goals. For example, Goldreich and Levin [38] in their work on cryptographic hardcore predicates for any one way function introduce the goal of *locally list decoding*, that is, finding all messages whose codeword is close to a given corrupted codeword while reading only a small number of entries in the corrupted codeword. The need for locality emerge from their use of the Hadamard code –a code with exponential encoding length– while working in an unorthodox input model where corrupted codewords are given by black box access rather than being transmitted. Goldreich and Levin [38] provide a local list decoding algorithm for the Hadamard code, presenting the first efficient (local or non-local) list decoding algorithm. Following [38] and Sudan's [81] list decoding algorithm for the Reed-Solomon codes, Sudan-Trevisan-Vadhan [83] presented a local list decoding algorithm for the Reed-Muller codes.

- **Local decoding/self correcting in constant query complexity.** Other applications in complexity and cryptography [9–12,22] achieve stronger locality for the Hadamard codes and for Reed-Muller based codes, in which: recovering each single message bit is done while reading only a *constant* number of corrupted codeword entries. Explicit treatment of such locally decodable codes appear in [60] and in subsequent works; see a survey in [85]. A related goal is to *locally self correct*, that is, to recover the correct value for a single *codeword entry* while reading only a *constant* number of corrupted codeword entries. The abovementioned locally decodable codes –except those appearing in [12]– are also locally self correctable.

In terms of *algebraic structure*, for the vast majority of error correcting codes, the underlying algebraic structure is a *field*. For example, this is the case in all above codes, and more generally, in all *linear codes* (which by definition are codes whose codewords form a subspace of a vector space over some field), as well as in almost all known non-linear codes. An exception are *group codes* [58,79] – a class of codes extending the class of linear codes by allowing codewords to form a *subgroup* of a group G^n (rather than a subspace of a vector space over a field).

The performance of linear codes, and, more generally, group codes, depends on properties of the underlying group G. In particular, their alphabet size is at least the size of the smallest (non-trivial) subgroup of G. For example, group codes with G a cyclic group of prime order p, have alphabet size p. Group codes (and linear codes) are therefore of interest primarily when the group G (field $\mathbb{F}$) is of *small order* (or has small order subgroups).

Our Work

In this work we introduce a new class of error correcting codes: *Multiplication codes (MPC)*, and develop decoding algorithms for them, showing they achieve desirable combinatorial and algorithmic properties, including: *binary alphabet*, *constant distance* together with efficient *local list decoding* and *local self correcting* algorithms for codes of exponential encoding length, efficient *decoding in random noise* model for codes of polynomial encoding length, and (non-polynomial time) *decoding in adversarial* noise model for codes of *linear encoding length.*

Our results give the first (asymptotic family of) codes achieving constant rate and distance, while having large groups as their underlying algebraic structure.[1] Likewise, our results give the first (asymptotic families of) codes achieving any of the algorithmic properties of being uniquely decodable, list decodable, locally list decodable or locally self correctable, while having large groups as their underlying algebraic structure.

Our techniques and algorithms are applicable beyond the scope MPC: (1) Our local list decoding algorithm is applicable to any Fourier concentrated and recoverable codes.[2] In particular, our algorithm gives a list decoding algorithm for the *homomorphism codes* over any finite abelian group and into the complex numbers. (2) We provide a *soft local error reduction algorithm* for ABNNR codes [7] concatenated with binary codes. This algorithm offers an alternative to Forney's GMD decoding approach for those concatenated codes.

In the following we first define the class of MPC codes. Next, we present the algorithmic and combinatorial results we achieve for MPC codes. We then elaborate on some applications of our algorithms beyond the scope of MPC. Finally, we mention new techniques we developed for our algorithms.

The Class of Multiplication Codes

Historically, we first defined our MPC codes in the context of our study of cryptographic hardcore predicates for number theoretic one-way functions [3]. We defined there [3] codes $\mathcal{C}^P$ encoding messages $m \in \mathbb{Z}_N$ by values $P(x)$ for $P\colon \mathbb{Z}_N \to \{\pm 1\}$ a Boolean predicate. Specifically, the codeword encoding m is:

$$C(m) = (P(m \cdot 1), P(m \cdot 2), \ldots, P(m \cdot N))$$

where $m \cdot i$ is multiplication modulo N. We gave there [3] a local list decoding algorithm for such codes $\mathcal{C}^P$, provided P is "Fourier well concentrated" (namely, most of the energy of its Fourier transform is concentrated on a small set of significant

[1]By *large groups* we refer to groups having no (non-trivial) small subgroups, where "small" is, say, constant size, or size logarithmic in information rate. For example, the additive group $\mathbb{Z}_N$ of integers modulo N for $N = pq$ a product of two large primes $p, q = \Omega(\sqrt{N})$ is a large group with respect to the message space $\mathbb{Z}_N^*$ (which has information rate $\log N - o(1)$).

[2]A code is *concentrated* if –when identifying codeword with complex functions over a finite abelian group– its codewords can be approximated by a sparse Fourier representation; a code is *recoverable* if a codeword can be efficiently recognized when given one of its significant Fourier coefficients.

Fourier coefficients α with $gcd(\alpha, N) \leq poly(\log N)$). This algorithm was inspired by the Goldreich-Levin [38] local list decoding algorithm for the Hadamard codes.

Going beyond the context of cryptographic hardcore predicates, we extend the above definition in two ways. First, to include *codes of good rate*, we extend the definition of MPC to allow restrictions of the above codewords to a subset $S = s_1, \ldots, s_n \subseteq \mathbb{Z}_N$ of their entries. For example, taking S to be a random subset of $\mathbb{Z}_N$ of size $O(\log N)$ the resulting code $\mathcal{C}^{P,const}$ has codewords

$$C^{const}(m) = (P(m \cdot s_1), P(m \cdot s_2), \ldots, P(m \cdot s_n))$$

For this example, we show that for a good choice of the predicate P, the code $\mathcal{C}^{P,const}$ has constant rate and distance.

Furthermore, to include also codes whose *underlying algebraic structure is any abelian group* we further extend the definition of MPC as follows:

Definition 6.1 (Multiplication codes (MPC))**.** *For any abelian group G, an alphabet controlling function $P \colon \mathbb{C} \to \mathbb{C}$, and a set $S = \{s_1, \ldots, s_n\} \subseteq G$ of indexes to codeword entries,*[3] *we denote by (G, P, S) the MPC code that encode messages $m \in G$ by codewords*

$$C(m) = (P(\chi_m(s_1)), P(\chi_m(s_2)), \ldots, P(\chi_m(s_n)))$$

where $\chi_m \colon G \to \mathbb{C}$ is the homomorphism corresponding to m.[4]

Example 6.2. *Two examples of binary Multiplication codes for the group $\mathbb{Z}_N$ are the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ with codeword encoding messages m in $\mathbb{Z}_N$ defined by*

$$C_m = (\mathsf{half}_N(1 \cdot m), \mathsf{half}_N(2 \cdot m), \mathsf{half}_N(3 \cdot m), \ldots, \mathsf{half}_N(N \cdot m))$$

where $\mathsf{half}_N \colon \mathbb{Z}_N \to \{0, 1\}$ is defined by

$$\mathsf{half}_N(x) = \begin{cases} 1 & \textit{iff } \min\{x, N - x\} \leq N/4 \\ 0 & \textit{otherwise} \end{cases}$$

and the codes $\mathcal{C}^{\mathsf{half},const} = (\mathbb{Z}_N, \mathsf{half}_N, R_N)$ with codeword encoding messages m in $\mathbb{Z}_N$ defined by

$$C_m^{const} = (\mathsf{half}_N(r_1 \cdot m), \ldots, \mathsf{half}_N(r_n \cdot m))$$

where $R_N = \{r_1, \ldots, r_n\}$ is a random subset of $\mathbb{Z}_N$ of size $O(\log N)$.

We use the terminology "*MPC code for G*", when we want to emphasize that G is the underlying group. Similarly, when addressing asymptotic families of MPC codes with growing underlying groups G_N, we use the terminology "*MPC code for $\{G_N\}_N$*".

[3]More generally, we also consider MPC where $S \subseteq G \times \ldots \times G$.

[4]The homomorphism $\chi_m \colon G \to \mathbb{C}$ corresponding to m is defined as follows. For $G = \mathbb{Z}_N$, $\chi_m(s) = e^{i2\pi ms/N}$. For $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, $\chi_{m_1,\ldots,m_k}(s_1, \ldots, s_k) = e^{i2\pi \sum_{j=1}^{k} m_j s_j / N_j}$. In general, for G a multiplicative group with a generating set $\{g_1, \ldots, g_k\}$ where g_j has order N_j, $\chi_{\prod_{j=1}^{k} g_j^{m_j}}(\prod_{j=1}^{k} g_j^{s_j}) = e^{i2\pi \sum_{j=1}^{k} m_j s_j / N_j}$.

MPC codes as a class of codes extends the class of (abelian) group codes and the class of linear codes.

Remark 6.3. *In retrospect, any MPC code can be thought of as being defined by a group code $\mathcal{C} \subseteq \Sigma_1^n$ together with an alphabet reduction function $P\colon \Sigma_1 \to \Sigma_2$ mapping codewords $(C_1, \ldots, C_n) \in \Sigma_1^n$ of the group code $\mathcal{C}$ into codewords $(P(C_1), \ldots, P(C_n)) \in \Sigma_2^n$ of the MPC. The challenge is to find group codes $\mathcal{C}$ and alphabet reduction functions P that result in an MPC codes achieving good combinatorial and algorithmic properties.*

The alphabet reduction method method we present here is useful even for very large alphabet. *For example, we use it to reduce the alphabet of homomorphism codes (i.e., an alphabet which is as large as the message space). This is in contrast to the well known* codes concatenation *alphabet reduction method, which is useful only in small alphabet settings.*[5]

Multiplication Codes Achieving Good Properties

Our main result is presenting three (asymptotic families of) MPC codes for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$ all achieving binary alphabet and constant distance[6] and the following algorithmic properties and encoding length: (1) Codes of encoding length N which are *locally list decodable* and *locally self correctable.* (2) Codes of *encoding length polynomial in* $\log N$, which are decodable in *random noise model* in time polynomial in $\log N$, and are decodable in *adversarial noise model* in time $N^{2\varepsilon} poly(\log N)$ (for ε the fraction of flipped bits). (3) Codes of *constant rate and distance.*

We use the notation $(k, n, d)_q$-code referring to codes of message space size 2^k, encoding length n, normalized Hamming distance d, and alphabet size q.

Theorem 6.4 (MPC codes for $\{\mathbb{Z}_N\}_{N\in\mathbb{N}}$)**.** *We present three (asymptotic families of) MPC codes for groups $\mathbb{Z}_N$ of growing sizes N:*

1. **Codes with local algorithms***: $(\Theta(\log N), N, \Theta(1))_2$-codes, which are efficiently* locally list decodable *and* locally self correctable. *The local list decoding algorithm we present has query complexity and running time poly*$(\log N)$*. The local self correcting algorithm we present has query complexity $\Theta(1)$ and running time poly*$(\log N)$*. The input to the local self correcting algorithm is restricted to entries $s \in \mathbb{Z}_N^*$. Both algorithms are randomized.*[7]

[5]In *codes concatenation*, alphabet Σ_1 is reduced to a smaller alphabet Σ_2 by encoding each symbol $\sigma \in \Sigma_1$ using a code of alphabet Σ_2. This method is of interest only when the alphabet Σ_1 is considerably smaller than the message space size (otherwise finding a good code for the message space Σ_1 is as hard as finding a good code for the original message space). That is, concatenation is only useful when the alphabet in not too large to begin with.

[6]The code distance is the minimum relative Hamming distance between any two of its codewords, where the relative Hamming distance is the fraction of entries on which they differ.

[7]In the local list decoding algorithm success probability is taken only over the random coins of the algorithm, namely, it is independent of the input; success probability $1-\rho$ is achieved in time $poly(\log N, \ln(1/\rho))$. In the local self correcting algorithm we present in this chapter, success probability is taken both over the input and over the random coins of the algorithm; success probability $1-\rho$ is achieved in query complexity $poly(1/\rho)$ and running time $poly(\log N, 1/\rho)$. This algorithm

2. **Polynomial rate codes with efficient decoding**: $(\Theta(\log N), poly(\log N), \Theta(1))_2$-codes, *which are efficiently decodable in the random noise model. The decoding algorithm we present runs in time* $poly(\log N)$. *Furthermore, these codes are decodable in adversarial noise model in time* $N^{2\varepsilon} poly(\log N)$ *for* ε *the fraction of flipped bits.*

3. **Codes of constant rate & distance**: $(\Theta(\log N), \Theta(\log N), \Theta(1))_2$-*codes.*

Remark 6.5. *For linear codes, local decodability is implied by any local self correcting algorithm robust against the changes of basis (because there is a basis change transforming the code to* systematic*, where message bits appears as part of the codeword). In contrast, for* non-*linear codes —as are MPC— a systematic representation does not necessarily exist, and a relation between local self correcting and local decoding is not known.*

Extending the above results to other groups, we present MPC codes for any finite abelian group with a constant size generating set. These codes achieve parameters as in the above theorem, albeit with alphabet size $2^{O(k)}$ for $k = \Theta(1)$ the generating set size. The algorithms we present receive a description of the underlying group by its generators and their orders as part of the input.

Theorem 6.6 (MPC for groups of small generating sets). *Let* G *be a finite abelian group with a generating set* $\{g_1, \dots, g_k\}$ *of size* $k = O(1)$. *Denote* $N_1, \dots, N_k$ *the orders of* $g_1, \dots, g_k$, *and denote by* $N = \prod_{i=1}^{k} N_i$ *the size of* G. *We present three MPC codes for groups* G*:*

1. **Codes with local algorithms**: $(\Theta(\log N), N, \Theta(1))_{2^{O(k)}}$-*codes, which are efficiently* locally list decodable *and* locally self correctable. *The local list decoding algorithm we present has query complexity and running time* $poly(\log N)$. *The local self correcting algorithm we present has query complexity* $\Theta(1)$ *and running time* $poly(\log N)$. *The input to the local self correcting algorithm is restricted to entries* $s \in G^*$. *Both algorithms are randomized.*[8]

2. **Polynomial rate codes with efficient decoding**: $(\Theta(\log N), poly(\log N), \Theta(1))_{2^{O(k)}}$-*codes, which are efficiently decodable in the random noise model. The decoding algorithm we present runs in time* $poly(\log N)$. *Furthermore, these codes are decodable in adversarial noise model in time* $\sum_{i=1}^{k} N_i^{2\varepsilon} poly(\log N_i)$ *for* ε *the fraction of flipped bits.*

3. **Codes of constant rate & distance**: $(\Theta(\log N), \Theta(\log N), \Theta(1))_{2^{O(k)}}$-*codes.*

can be improved to achieve success probability depending *only* on the random coins of the algorithm.

[8]In the local list decoding algorithm success probability is taken only over the random coins of the algorithm, namely, it is independent of the input; success probability $1 - \rho$ is achieved in time $poly(\log N, \ln(1/\rho))$. In the local self correcting algorithm we present in this chapter, success probability is taken both over the input and over the random coins of the algorithm; success probability $1 - \rho$ is achieved in query complexity $poly(1/\rho)$ and running time $poly(\log N, 1/\rho)$. This algorithm can be improved to achieve success probability depending *only* on the random coins of the algorithm.

Remark 6.7. *1. When all generators have the same order, i.e., $N_1 =, \ldots, = N_k$, the alphabet of the codes from the above theorem is of size 2^k. In particular, for cyclic groups, the alphabet is binary, i.e., of size* 2.

2. Binary *MPC codes achieving properties as the above can be obtained by concatenating the above codes with a good binary code. Such a bianry code can be efficiently found, e.g., by exahustive search, since the alphabet size is constant.*

3. The list decoding algorithm returns a list of polynomial size. This is a direct result of the polynomial running time bound.

4. Explicit constructions can be derived from small biased sets for G.

Remark 6.8. *An application of the local self correcting algorithm is for improving the running time complexity of the list decoding algorithm. This is by reducing the noise in the given corrupted codeword via locally self correcting each of the queries the decoding algorithm makes to the corrupted codeword. This improvement is possible in the noise range $\varepsilon \leq 0.15$ where the local self correction algorithm is applicable, and improves the running time dependency on the noise ε by a constant, e.g., by a factor of roughly 10^5 for $\varepsilon = 0.15$.*

New Techniques

Decoding via learning (in query and subset access models). For our decoding algorithms we develop a *decoding via learning approach*, where we identify codewords with complex functions over a finite abelian group (or, restrictions of such functions to a subset of their entries), and decode by first finding the significant Fourier coefficients of the given corrupted codeword, and then mapping the significant Fourier coefficients to the messages of the close codewords.

Finding significant Fourier coefficients: For codes whose codewords are functions over finite abelian groups, we find their significant Fourier coefficients using our SFT algorithm. For codes whose codewords are restrictions of such functions to a subset of the finite abelian group, we develop new algorithms for finding their significant Fourier coefficients. These algorithms find significant Fourier coefficients of a signal when given values of the signal on a (carefully designed) predetermined set of entries and where values are corrupted by (random or adversarial) noise.

Mapping significant Fourier coefficients to messages of close codewords: For linear codes (or, more generally, group codes) the significant Fourier coefficient immediately map to the messages encoded by the close codewords. For the MPC codes that we present, such a map is not always immediate. Nevertheless, we show that such a map exists and can be computed efficiently.

The decoding via learning is applicable to any Fourier concentrated and recoverable codes, that is, codes whose codewords can be approximated by few significant coefficients in their Fourier transform and such that there is an efficient mapping from a Fourier coefficient to all codewords for which it is significant.

Self correcting via testing. Our local self correcting algorithm is composed of two parts: (1) A reduction from the self correcting problem to the property testing problem of distinguishing between signals with high and low Fourier coefficients in a large interval, and (2) An algorithm for solving the property testing problem.

Both the reduction algorithm and the property testing algorithm make only a constant number of queries to the given corrupted codeword.

Soft Error Reduction for Concatenated ABNNR Codes We present an alternative to Forney's Generalized Minimum Distance (GMD) methodology [33] for decoding concatenated ABNNR codes [7] with binary codes. Our algorithm takes a *soft decoding approach* to replace the hard decision in the Forney's methodology. The algorithm we present is *local*, namely, each bit of message symbol can be found with a d^2 queries to the codewords for d the degree of the expander graph underlying the ABNNR code.

Other Related Works

The idea of *list decoding* —that is, finding the list of all codewords close to the given corrupted codeword— was proposed in the 1950's by Elias [28] and Wozencraft [89] for dealing with situations where the noise is too great to allow unique decoding. Efficient list decoding algorithms followed many years later, starting with the Goldreich and Levin [38] list decoding algorithm for the Hadamard code, which is a code of exponentially small rate (*i.e.*, codeword length is exponential in message length). A few years later, Sudan [81] presented the first polynomial time list decoding algorithm for codes of polynomial rate: the Reed-Solomon codes. In following years more list decoding algorithms were presented, including improved list decoding algorithm for Reed-Solomon codes [48], and polynomial time list decoding algorithms for: Reed-Muller codes [83], Algebraic Geometric codes [48, 77], Chinese Remainder codes [39], certain concatenated codes [49], graph based codes [45], "XOR lemma based" codes [84] (codes defined there), and folded Reed-Solomon codes [47]. Following our work [3], Grigorescu *et.al.* [43] presented a list decoding algorithm for homomorphism codes whose domain and range are both any finite abelian group, their algorithm correct errors up to large relative Hamming distances.

The list decoding algorithms of Goldreich-Levin [38], Sudan-Trevisan-Vadhan [83], Trevisan [84] and Grigorescu *et.al.* [43] are *local* list decoding algorithms, where the query complexity is polynomial in $\log N$ for N the codeword length and in the distance parameter in [38, 43, 83], and it is of the order of $\sqrt{N}$ and exponential in the distance parameter in [84].

Chapter Organization

The rest of this chapter is organized as follows. In section 6.2 we give a detailed overview of the techniques and algorithms we develop for proving our results. In sections 6.3-6.8 we give formal statement of our algorithms and complete proofs which were omitted from the overview. In details: We present our decoding via learning approach and show how it applies to concentrated and recoverable codes in section 6.3. We analyze the combinatorial properties of our codes for $\mathbb{Z}_N$ in section 6.4. We present our local self correcting algorithm for codes for $\mathbb{Z}_N$ in section 6.5. We show how to obtain codes of linear encoding length from codes of super linear encoding length while preserving the code distance in section 6.6. We present codes for groups of small generating sets in section 6.7. We present our soft error reduction algorithm for concatenated ABNNR codes in section 6.8.

6.2 Overview of Results and Techniques

In this section we give a detailed overview of the techniques and algorithms we develop for proving our results. We start by presenting our decoding via learning approach in section 6.2.1, and show how it implies list decoding algorithms for MPC codes for $\mathbb{Z}_N$ and for homomorphism codes over arbitrary finite abelian groups. We then focus on MPC codes for groups $\mathbb{Z}_N$, presenting distance bound, and local self correcting algorithm for them in sections 6.2.2-6.2.3. Next we address MPC codes for groups of small generating sets in section 6.2.4. We then show how to obtain MPC codes of linear encoding length and constant distance from the exponential length codes we already constructed in section 6.2.5. Finally, we describe our soft error reduction algorithm in section 6.2.6.

6.2.1 List Decoding via Learning

In this section we present our decoding via learning approach, and show how it implies list decoding algorithms for MPC codes for $\mathbb{Z}_N$ and for homomorphism codes over arbitrary finite abelian groups.

A code is *locally list decodable* if there is an algorithm that returns all codewords in a given radius from a given a corrupted codeword w, while reading only a number of entries in w poly logarithmic in its length.

Definition 6.9 (List decoding)**.** *We say that a code is* (ε, T)-list decodable *if there is an algorithm that, given a description of the group G by its generators and their orders, query access*[9] *to a corrupted codeword w and a noise parameter ε outputs all codewords within relative Hamming distance at most ε from w (with probability at least* $2/3$*) and its running time is at most T. When T is smaller than the encoding length, the algorithm does not read the entire corrupted codeword, and we say that the code is* (ε, T)-locally *list decodable.*

Remark 6.10. *1. The success probability of* $2/3$ *in the local list decoding or local self correcting algorithms can be amplified to* $(\frac{2}{3})^c$ *by standard methods incurring a running time and query complexity increase by a factor of c.*

[9]We say that an algorithm is given *query access* to $w \in \Sigma^S$ if it can ask and receive the value of w on each entry $s \in S$ in unit time.

2. *In the list decoding algorithm, the bound T on the running time in particular implies a bound T on the length of the outputted list. When we want to emphasize a (possibly, tighter) bound on the list size, we say that the code is (ε, ℓ, T)*-list decodable *for ε, T as in Definition 6.9 above, and ℓ a bound on the size of the outputted list.*

3. *We use the terminology* list decoding in ℓ_2 distance *to describe an algorithm that, given a corrupted codeword w and a distance parameter ε, returns all codeword in the ball $B_{\mathcal{C},\ell_2}(w, \varepsilon) = \{C \in \mathcal{C}, \|C - w\|_2^2 \leq \varepsilon\}$ around w according to the ℓ_2 rather than the Hamming distance measure.*

Decoding via Learning in Query Access

Our list decoding algorithm follows our *decoding via learning approach*, where a key algorithmic component is finding the significant Fourier coefficients of the given corrupted codeword in polynomial time (*i.e.*, in time polynomial in the information rate).

Abstractly, our list decoding algorithm is applicable to any "concentrated" and "recoverable" code over a "learnable domain"; defined as follows. Let $\mathcal{C}$ be a code with codewords $C_x \colon G \to \mathbb{C}$ identified with complex functions over a finite abelian group G. We say that $\mathcal{C}$ is *concentrated* if its codewords can be approximated by a small number of significant coefficients in their Fourier representation. We say that $\mathcal{C}$ is *recoverable* if there is an efficient algorithm mapping each Fourier coefficient α to a short list of codewords for which α is a significant Fourier coefficient. We say that G is *over a learnable domain* if there is an algorithm that, finds the significant coefficients of functions f over G when given a description of G, and threshold τ and query access to f, and its running time is polynomial in $\log|G|, 1/\tau$.

Definition 6.11. *Let $\mathcal{G} = \{G_N\}_{N\in\mathbb{N}}$ be a family of finite abelian groups indexed by integers N. Let $\mathcal{C} = \{C_{N,x} \colon G_N \to \mathbb{C}\}_{N\in\mathbb{N},x\in D_N}$ be a code with codewords $C_{N,x}$ of length $|G_N|$ that encode elements x from a message space D_N.*

1. **Fourier concentration.** *Let $\mathcal{F} = \{f_N \colon G_N \to \mathbb{C}\}_{N\in\mathbb{N}}$ be a family of functions. We say that $\mathcal{F}$ is* Fourier concentrated *if for every $\varepsilon > 0$ there exists a small set Γ of characters s.t. $|\Gamma| \leq poly(\log|G_N|/\varepsilon)$ and*

$$\|\sum_{\chi_\alpha\in\Gamma} \widehat{f_N}(\alpha)\chi_\alpha - f_N\|_2^2 \leq \varepsilon$$

We say that a code $\mathcal{C} = \{C_{N,x} \colon G_N \to \mathbb{C}\}_{N\in\mathbb{N},x\in D_N}$ is Fourier concentrated *if its codewords are Fourier concentrated functions.*

2. **Fourier recoverable codes.** *For each code $\mathcal{C} = \{C_{N,x} \colon G_N \to \mathbb{C}\}_{N\in\mathbb{N},x\in D_N}$, threshold $\tau \in \mathbb{R}^+$ and character index $0 \neq \beta \in G_N$, denote by $\mathsf{InvHeavy}_{N,\tau,\beta}(\mathcal{C})$ the set of all messages $x \in D_N$ s.t. the β-Fourier coefficients of the codeword*

$C_{N,x}$ encoding x is τ-heavy (i.e., its squared magnitude at least τ)

$$\mathsf{InvHeavy}_{N,\tau,\beta}(\mathcal{C}) \overset{def}{=} \left\{ x \in D_N \mid \left| \widehat{C_{N,x}}(\beta) \right|^2 \geq \tau \right\}$$

We say that $\mathcal{C}$ is recoverable *if $\forall N \in \mathbb{N}, \tau \in \mathbb{R}^+, 0 \neq \beta \in G_N$, $\left|\mathsf{InvHeavy}_{N,\tau,\beta}(\mathcal{C})\right| \leq$ poly$(\log |G_N| / \tau)$. We say that $\mathcal{C}$ is* efficiently recoverable*, if there exists a recovery algorithm, that given $N, \tau, \beta \neq 0$, returns a list $L \supseteq \mathsf{InvHeavy}_{N,\tau,\beta}(\mathcal{C})$; and its running time is at most poly$(\log |G_N| / \tau)$.*

Remark. *If $\mathcal{C}$ is* efficiently recoverable*, then, in particular, $\mathcal{C}$ is recoverable, because a running time bound implies a bound on output size.*

3. **Learnable domain.** *We say that $\mathcal{G} = \{G_N\}_{N\in\mathbb{N}}$ is a* learnable domain *if qLCN with query access to $\mathcal{G}$ is in BPP. Namely, there is an algorithm that given N, τ and query access to any function $f_N \colon G_N \to \mathbb{C}$, outputs a list of length $O(1/\tau)$ that contains all τ-significant Fourier coefficients of f_N; and the running time of this algorithm is poly$(\log |G_N|/\tau)$.*

   Similarly, we say that $\mathcal{Q} = \{Q_{N,\tau} \subseteq G_N\}_{N\in\mathbb{N},\tau\in\mathbb{R}^+}$ is a learnable domain *w.r. to a code $\mathcal{C}$ if $(\mathcal{Q}, \mathcal{C})$-LCN is in BPP (see definition 5.14).*

The key property of concentrated codes is that received words w share a significant Fourier coefficient with all close codewords C_x. The high level structure of our list decoding algorithm is therefore as follows. First it runs an algorithm that finds all significant Fourier coefficients α of the received word w. Second for each such α, it runs the recovery algorithm to find all codewords C_x for which α is significant. Finally, it outputs all those codewords C_x. The running time of this list decoding algorithm is polynomial in $\log |G|$.

Theorem 6.12 (List decoding). *Let $\mathcal{C} = \{C_{N,x} \colon G_N \to \{0,1\}\}_{N\in\mathbb{N},x\in D_N}$ be a family of balanced[10] binary codes.*

1. **Combinatorial list decoding bound.** *If $\mathcal{C}$ is a Fourier concentrated and recoverable code, then for any $C_{N,x} \in \mathcal{C}$ and $\varepsilon \in [0, \frac{1}{2})$, the number of codewords in the ball of radius ε around $C_{N,x}$ is at most*

$$\left| \mathcal{B}_{\mathcal{C},Hamming}(C_{N,x}, \frac{1}{2} - \varepsilon) \right| \leq poly(\log |G_N| / \varepsilon)$$

   where $\mathcal{B}_{\mathcal{C},Hamming}(C_{N,x}, \frac{1}{2} - \varepsilon) = \left\{ C \in \mathcal{C} \mid \Delta(C, C_{N,x}) \leq \frac{1}{2} - \varepsilon \right\}$ is the set of codewords in the Hamming ball of radius $\frac{1}{2} - \varepsilon$ around $C_{N,x}$.

2. **Efficient list decoding algorithm.** *If $\mathcal{C}$ is a Fourier concentrated and* efficiently *recoverable code and $\mathcal{G} = \{G_N\}_{N\in\mathbb{N}}$ is a* learnable domain*, then $\mathcal{C}$ is $(\frac{1}{2} - \varepsilon, poly(\log |G_N| / \varepsilon))$-list decodable with for any $\varepsilon \in [0, \frac{1}{2})$.*

[10]We say that a code is *balanced* if for each codeword, the number of 0 values entries is roughly the same as the number of 1 values entries, that is, $\forall C_{N,x}$, $||\{ s \in G_N \mid C_{N,x}(s) = 1\}| - |\{ s \in G_N \mid C_{N,x}(s) = 1\}|| < \nu(\log N)$ for ν a negligible function.

Remark 6.13. 1. *The theorem extend to* non binary *codes as long as codewords have bounded ℓ_2-norm, $\|C\|_2 \leq 1$.*

2. *The theorem extends to the ℓ_2* distance. *That is, denote by $d = \min_{C,C' \in \mathcal{C}} \|C - C'\|_2^2$ the code distance with respect to the ℓ_2-distance, then (1) under the conditions of above theorem part 1, $|\mathcal{B}_{\mathcal{C},\ell_2}(C_{N,x}, d - \varepsilon)| \leq poly(\log |G_N| / \varepsilon)$ for all $\varepsilon \in [0, d)$ and $C_{N,x} \in \mathcal{C}$ where $\mathcal{B}_{\mathcal{C},\ell_2}(C_{N,x}, d - \varepsilon) = \{C \in \mathcal{C} \mid \|C - C_{N,x}\|_2^2 \leq d - \varepsilon\}$ is the set of codewords in the ℓ_2 ball of radius $d - \varepsilon$ around $C_{N,x}$, and (2) under the conditions of above theorem part 2, $\mathcal{C}$ is $(d - \varepsilon, poly(\log |G_N| / \varepsilon))$-list decodable with respect to the ℓ_2-distance for any $\varepsilon \in [0, d)$.*

3. *The theorem extend to* unbalanced codes *provided that $\varepsilon \in [0, d - bias)$ for $bias = \max_{N,x} \left|\widehat{C}_{N,x}(0)\right|$ and d is the code distance in the norm of interest (ℓ_2 norm of relative Hamming norm).*

4. *We already proved that any finite abelian group G is a learnable domain (this is by our SFT algorithm discussed in chapter 3). Therefore, for codes with codewords identified as functions over a finite abelian group G, we only need to prove that the code is concentrated and recoverable.*

Decoding via Learning in Subset Access

Fourier concentrated and recoverable codes $\mathcal{C} \subseteq \{C : G \to \mathbb{C}\}$ have encoding length $|G|$ which is exponential in their information rate.[11] Better rate codes $\mathcal{C}_{|S}$ can be obtained by *restricting* all codewords of $\mathcal{C}$ to a subset $S \subseteq G$ of their entries. This results in codes achieving *polynomial encoding length* when taking S to be of size $|S| = poly(\log |G|)$.

We show that restrictions $\mathcal{C}_{|S}$ of Fourier concentrated and recoverable codes are efficiently decodable in random noise models, as long as S has the following property: There is an efficient algorithm finding the significant Fourier coefficients of any codeword $C \in \mathcal{C}$ when given access —only to the entries in S— to a corruption of C in random noise models.

More generally, the above holds for any noise model: For any noise model M, we say that *M-Faulty $(S, \mathcal{C})$-LCN is in solvable in time T* if there is an algorithm finding the significant Fourier coefficients of any codeword $C \in \mathcal{C}$ when given access —only to the entries in S— to a corruption of C in noise model M; and the running time of the algorithm is T. We show that if M-Faulty $(S, \mathcal{C})$-LCN is solvable in time T, then $\mathcal{C}$ is decodable in time $T \cdot poly(\log |G|)$. Namely, there is an algorithm that given a codeword corrupted in noise model M, outputs the encoded message, and its running time is $T \cdot poly(\log |G|)$.

Theorem 6.14. *Let M be a noise model, $\mathcal{C} = \{C : G \to \mathbb{C}\}$ a Fourier concentrated and recoverable code, and $S \subseteq G$ a subset s.t. M-Faulty $(S, \mathcal{C})$-LCN is solvable in time*

[11]The message space of recoverable codes is at most $|G| \cdot poly \log |G|$ (because each Fourier coefficients is mapped to a list of $poly \log |G|$ codewords); namely, their information rate is $\Theta(\log |G|)$. The encoding length $|G|$ is thus exponential in the information rate.

T. Then $\mathcal{C}$ is decodable in time $T \cdot poly(\log|G|)$.

Proof. The proof of this theorem is analogous to the proof of Theorem 6.12, while replacing the use of the SFT algorithm with the algorithms for finding significant Fourier coefficients of M-faulty functions in the S-access model. Details omitted. □

We show that there are polynomial size sets S, $|S| = poly \log|G|$, s.t. M-Faulty $(S, \mathcal{C}_N)$-LCN is efficiently solvable, for M the random noise model (see chapter 5, Theorem 5.15). Therefore, any Fourier concentrated and recoverable codes $\mathcal{C}$ can be restricted to codes $\mathcal{C}_{|S}$ of polynomial encoding length that are efficiently decodable in random noise models.

Corollary 6.15. *Let M be a noise model independently flipping each codeword symbol uniformly at random with probability ε for $\varepsilon = O(1)$ sufficiently small. Let $\mathcal{C} = \{C : G \to \mathbb{C}\}$ a Fourier concentrated and recoverable code. There exists a subset $S \subseteq G$ of size $|S| = poly(\log|G|, 1/\varepsilon)$ s.t. $\mathcal{C}_{|S}$ is decodable in noise model M in time $poly(\log|G|)$.*

For M an adversarial noise model flipping ε-fraction of the codewords entries, we show there are linear size sets S, $|S| = \Theta(\log|N|)$, s.t. M-Faulty $(S, \mathcal{C}_N)$-LCN is solvable in time $N^{2\varepsilon} \cdot poly \log N$ (see chapter 5, Theorem 5.15). Therefore, any Fourier concentrated and recoverable codes $\mathcal{C}$ can be restricted to codes $\mathcal{C}_{|S}$ of linear encoding length that are decodable in adversarial noise model in time $N^{2\varepsilon} \cdot poly \log N$.

Corollary 6.16. *Let M be an adversarial noise model flipping ε-fraction of the codewords entries for $\varepsilon = O(1)$ sufficiently small. Let $\mathcal{C} = \{\mathbb{Z}_N : G \to \mathbb{C}\}$ a Fourier concentrated and recoverable code. There exists a subset $S \subseteq \mathbb{Z}_N$ of size $|S| = \Theta(\log N) poly(1/\varepsilon)$ s.t. $\mathcal{C}_{|S}$ is decodable in noise model M in time $N^{2\varepsilon} \cdot poly \log N$.*

Remark 6.17. *To be more precise, the above theorem and corollaries actually address* asymptotic families *of Fourier concentrated and recoverable codes and of subsets.*

Applying the Decoding via Learning Approach

We present examples of Fourier concentrated and efficiently recoverable codes. For any finite abelian groups G, *homomorphism codes* over G into the complex are Fourier concentrated and efficiently recoverable codes. This is because the Fourier spectrum of their codewords has all its energy on a single element, the elements corresponding to the encoded message. We conclude therefore that homomorphism codes are locally list decodable.

Proposition 6.18. *For any finite abelian group G, the homomorphism code over G into the complex $\mathcal{C}_G = \{\chi_m : G \to \mathbb{C} \mid \chi_m(x+y) = \chi_m(x)\chi_m(y)\}$ are Fourier concentrated and efficiently recoverable.*

Corollary 6.19. *For any finite abelian group G, the homomorphism code over G into the complex $\mathcal{C}_G = \{\chi_m : G \to \mathbb{C} \mid \chi_m(x+y) = \chi_m(x)\chi_m(y)\}$ are $(2-\varepsilon, poly(\log|G|, 1/\varepsilon))$-locally list decodable with respect to the ℓ_2 distance.*

Remark 6.20. *1. Homomorphism codes over G and into the complex can be represented by a Multiplication code (G, P_G, G) by setting the alphabet controlling function $P_G \colon G \to \mathbb{C}$ to be $P_G(x_1, \ldots, x_m) = e^{i \sum_{j=1}^{k} \frac{2\pi}{N_j} x_j}$ for $(x_1, \ldots, x_m) \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ where $N_1, \ldots, N_k$ are the orders of the generators of G given as input to the algorithm.*

2. *An equivalent way to think of the SFT algorithm is as a local list decoding algorithm (in the ℓ_2 distance) for the homomorphism codes. This is because a codeword C_m of the homomorphism code is close to a corrupted codeword w in the ℓ_2-distance iff the m-th Fourier coefficient of w is heavy, specifically, $\|C_m - w\|_2^2 = 2(1 - |\widehat{w}(m)|)$.*[12]

3. *The SFT algorithm also provides a local list decoding algorithm in the* relative Hamming distance*, because the list of codewords at distance at most ε from w in the ℓ_2-norm contains the list of all codewords at distance at most 4ε from w in the relative Hamming distance.*

For the additive groups $\mathbb{Z}_N$ of integers modulo N, we show that Multiplication codes $(\mathbb{Z}_N, P_N, \mathbb{Z}_N)$ are concentrated and efficiently recoverable if the functions P_N are "well concentrated" and efficiently computable,[13] where we say $\mathcal{P}$ is *well concentrated* if P_N can be approximating by a *poly* $\log N$ number of coefficients in its Fourier transform, and all these coefficients have small greatest common divisor with N. An example of an efficiently computable and well concentrated function is the function half_N presented in Example 6.2. We conclude therefore that the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ presented in Example 6.2 are locally list decodable. These results generalize to any finite abelian cyclic groups.

Definition 6.21 (Well concentrated)**.** *For any family of functions $\mathcal{P} = \{P_N \colon \mathbb{Z}_N \to \mathbb{C}\}_{N \in \mathbb{N}}$. We say that $\mathcal{P}$ is* well concentrated *if $\forall N \in \mathbb{N}, \varepsilon > 0$, exists a small set $\Gamma \subseteq \mathbb{Z}_N$ of Fourier coefficients s.t. (i) $|\Gamma| \leq poly(\log N/\varepsilon)$, (ii) $\|\sum_{\alpha \in \Gamma} \widehat{P_N}(\alpha)\chi_\alpha - P_N\|_2^2 \leq \varepsilon$, and (iii) for all $\alpha \in \Gamma$, $gcd(\alpha, N) \leq poly(\log N/\varepsilon)$ (where $gcd(\alpha, N)$ is the greatest common divisor of α and N).*

Theorem 6.22. *Let $\mathcal{P}$ be a family of well concentrated functions, then the Multiplication codes $\mathcal{C}^{\mathcal{P}} = \{(\mathbb{Z}_N, P_N, \mathbb{Z}_N)\}_{N \in \mathbb{N}}$ are Fourier concentrated and recoverable. If $\mathcal{P}$ are efficiently computable, then $\mathcal{C}^{\mathcal{P}}$ is also efficiently recoverable.*

[12]The correspondence $\|C_m - w\|_2^2 = 2(1 - |\widehat{w}(m)|)$ between ℓ_2-distance and Fourier coefficients is derived as follows. By Parseval Identity $\|C_m - w\|_2^2 = \sum_\alpha \left|\widehat{C_m - w}(\alpha)\right|^2$, which is equal to $\sum_\alpha \left|\widehat{C_m}(\alpha) - \widehat{w}(\alpha)\right|^2$ due to the linearity of the Fourier transform. Since $\widehat{C_m}(\alpha) = 1$ iff $\alpha = m$ and 0 otherwise, this is equal to $\sum_{\alpha \neq m} |\widehat{w}(\alpha)|^2 + |1 - \widehat{w}(m)|^2$. Rearranging this expression we get $\sum_\alpha |\widehat{w}(\alpha)|^2 + 1 - 2|\widehat{w}(m)|$. By Parseval Identity $\sum_\alpha |\widehat{w}(\alpha)|^2 = 1$ for w accepting values on the complex unit sphere (where recall that we identified the code alphabet with elements on the unit sphere). Thus we conclude that $\|C_m - w\|_2^2 = 2(1 - |\widehat{w}(m)|)$.

[13]$\mathcal{P} = \{P_N \colon \mathbb{Z}_N \to \{0,1\}\}_{N \in \mathbb{N}}$ is a family of *efficiently computable* functions if there is algorithm that given any $N \in \mathbb{N}$ and $x \in \mathbb{Z}_N$ outputs $P_N(x)$ in time *poly*$(\log N)$.

Lemma 6.23. *For any $N \in \mathbb{N}$, the function* $\mathsf{half}\colon \mathbb{Z}_N \to \{0,1\}$ *defined by* $\mathsf{half}_N(x) = 1$ *iff* $\min\{x, N-x\} \leq N/4$ *is efficiently computable and well concentrated.*

Corollary 6.24. *For any $N \in \mathbb{N}$, the codes $(\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ presented in Example 6.2 are is $(\frac{1}{2} - \varepsilon, poly(\log|G|, 1/\varepsilon))$-locally list decodable.*

We obtain polynomial rate codes along with decoding algorithms by taking *restriction* of the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ to subsets $S \subseteq \mathbb{Z}_N$ of their entries satisfying the following: There is an algorithm for finding the significant Fourier coefficients of codewords, when given access —only to entries in S— of a corrupted codeword. Specifically, taking the polynomial size subset S guaranteed by Theorem 5.15 in chapter 5, we obtain polynomial rate codes which are decodable in the random (adversarial) noise model in time $poly \log N$ ($N^{2\varepsilon} \cdot poly \log N$). These codes achieve properties as stated in parts 2 and 3 of theorem 6.4.

6.2.2 Self Correcting via Testing

We present our local self correcting algorithm for the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ for groups $\mathbb{Z}_N$ (where half_N is as defined in Example 6.2).

The local self correcting algorithm, given query access to a corrupted codeword C'_m and an entry location, reads only a *constant* number of entries of C'_m and returns the value on the given entry of the codeword C_m closets to C'_m (with high probability). This holds provided that the distance of the corrupted codeword C'_m from the closest codeword C_m is smaller than some fixed constant $\varepsilon = O(1)$. The success probability is taken both over the input and over the internal coins of the algorithm: for $(1-O(1))$-fraction of the codeword entries, the algorithm succeeds with high probability over its internal coins, whereas for $O(1)$ fraction of the codeword entries the algorithm fails.

We remark that this local list decoding utilizes particular properties of the functions half_N and is not generally applicable to other Multiplication codes. This is in contrast to the local list decoding algorithm which is applicable to all concentrated and efficiently recoverable codes.

Let $\mathcal{C}^{\mathsf{half}}_N = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ the MPC code as in Example 6.2. Denote $\tau_{max} = \max_{\alpha \in \mathbb{Z}_N} \left|\widehat{\mathsf{half}_N}(\alpha)\right|^2$, $\varepsilon \in [0, \tau_{max} - \frac{1}{4})$. We show that $\tau_{max} > \frac{1}{4} + \Theta(1)$.

Theorem 6.25 (Local self correcting)**.** *There is an algorithm that given $N \in \mathbb{N}$, $\varepsilon \in [0, \tau_{max} - \frac{1}{4})$, $\rho \in (0,1)$, an entry location $s \in \mathbb{Z}_N^*$ and query access to a corrupted codewords w s.t. $\Delta(w, C_m) < \varepsilon$ for $C_m \in \mathcal{C}^{\mathsf{half}}_N$, outputs the value $C_m(s)$ with probability at least $1-\rho$, while making $q \leq poly(1/(\tau_{max} - \frac{1}{4} - \varepsilon), 1/\rho)$ queries to w, and running in time $T \leq poly(\log|G|, 1/(\tau_{max} - \frac{1}{4} - \varepsilon), 1/\rho)$.*

Remark 6.26. *We extend the algorithm presented here to achieve success probability $1-\rho$ with logarithmic $\ln\frac{1}{\rho}$ query and running time dependency on ρ (rather than the polynomial dependency presented here). This extension is out of scope for this dissertation.*

The algorithmic core of our local self correcting algorithm is (1) a reduction from the self correcting problem of the codes $\mathcal{C}^{\mathsf{half}}$ to a property testing problem, and (2) an algorithm for solving the property testing problem. Both the reduction and the property testing algorithm that we present are computable with constant number of queries to the given corrupted codeword, and with running time polynomial in $\log N$; thus yielding a local self correcting algorithm with constant number of queries to the given corrupted codeword, and with running time polynomial in $\log N$.

The property testing problem to which we reduce the local self correcting task is the problem of distinguishing two cases regarding the Fourier weight of a given function $f\colon \mathbb{Z}_N \to \mathbb{C}$ over an interval $I \subseteq \mathbb{Z}_N$, where the Fourier weight of f over I is the sum of squared Fourier coefficients of f over I,

$$weight(I) \overset{def}{=} \sum_{\alpha\in I} \left|\widehat{f}(\alpha)\right|^2$$

The two cases two be distinguished are as follows.

- YES case: $weight(I) > \tau$
- NO case: $weight(\bar{I}) < \tau - O(1)$ for $\bar{I} \supseteq I$ an extension of of the interval I to a slightly larger interval

That is, given a description of an interval $I \subseteq \mathbb{Z}_N$, a threshold $\tau > 0$ and query access to a function $f\colon \mathbb{Z}_N \to \mathbb{C}$, the algorithms outputs 1 if $weight(I) > \tau$, outputs 0 if $weight(I) < \tau - O(1)$ and may output 0 or 1 otherwise.

We present below our reduction from the self correcting problem to the property testing problem, our algorithm for solving the property testing problem and its analysis.

Local Self Correcting the codes $\mathcal{C}^{\mathsf{half}}$: Algorithm & Analysis Overview

The high level structure of the local self correcting algorithm is as follows. The input is query access to a corrupted codeword C'_m, and an entry location $s \in \mathbb{Z}_N$. To locally self correct, our algorithm simulates query access to a corruption C'_{ms} of the codeword encoding $ms \in \mathbb{Z}_N$ in the code $\mathcal{C}^{\mathsf{half}}$, and outputs 1 iff the heaviest Fourier coefficient of C'_{ms} is in $[-\frac{N}{4}, \frac{N}{4}]$ (where we denote by $[-\frac{N}{4}, \frac{N}{4}]$ the set of all $\alpha \in \mathbb{Z}_N$ s.t. $\min\{\alpha, N-\alpha\} \leq \frac{N}{4}$).

We first argue why this gives the value of the s-th entry in the codeword closest to C'_m. By definition of the codes $\mathcal{C}^{\mathsf{half}}$, $C_m(s) = 1$ iff $ms \in [-\frac{N}{4}, \frac{N}{4}]$ (for ms the product of m and s in the group $\mathbb{Z}_N$). Fourier analysis of the codeword C_{ms} shows that its heaviest Fourier coefficients lie on the characters $\pm ms$. Thus, $C_m(s) = 1$ iff the heaviest Fourier coefficients of C_{ms} are within $[-\frac{N}{4}, \frac{N}{4}]$. This holds also for the corrupted codeword C'_{ms}, because the noise we consider is smaller than the difference between the heaviest and second heaviest Fourier coefficients implying that the heaviest Fourier coefficients of C_m and C'_m are located on the same characters.

We next elaborate on how to implement the steps of the algorithm with constant query complexity and $poly \log N$ running time, starting with explaining how we gain implicit access to C'_{ms}. Implicit access to C'_{ms} is gained by answering each query s'

with value $C'_m(ss')$ (*i.e.*, outputting the entry of C'_m indexed by $s \cdot s' \in \mathbb{Z}_N$). This gives a corrupted codeword C'_{ms} whose distance from C_{ms} is the same as the distance of the given corrupted codeword C'_m from C_m, because by definitions of codewords of $\mathcal{C}^{\mathsf{Half}}$ it holds that $C_{ms}(s') = C_m(ss')$. The complexity of answering each query to C'_{ms} is a single query to C'_m and *poly* $\log N$ running time for computing $ss' \mod N$.

We next explain how to decide whether or not the heaviest Fourier coefficient of C'_{ms} lies in $[-\frac{N}{4}, \frac{N}{4}]$ with $O(1)$ queries to C'_{ms} and $O(1)$ running time. Consider first the special case when we have the promise that the heaviest Fourier coefficient either lies within an interval $I \subseteq [-\frac{N}{4}, \frac{N}{4}]$ or it lies outside of $[-\frac{N}{4}, \frac{N}{4}]$, for an interval I satisfying the following: (i) I is *symmetric*[14], (ii) $|I| = O(N)$ is *sufficient small*, and (iii) I is *bounded away from* $\pm\frac{N}{4}$.[15] In this case, we show that the problem of deciding where the heaviest Fourier coefficient reduces to the problem of distinguishing the case $weight(I) > \frac{1}{2} + O(1)$ from the case $weight([-\frac{N}{4}, \frac{N}{4}]) < \frac{1}{2} - O(1)$; where for each interval $[a, b]$ in $\mathbb{Z}_N$ we denote by $weight([a, b])$ the sum of squared Fourier coefficients of C'_{ms} over $[a, b]$, that is, $weight([a, b]) = \sum_{\alpha \in [a,b]} \left|\widehat{C'_{ms}}(\alpha)\right|^2$.[16] We distinguish these two cases using adaptation of our techniques in [3] for estimating approximate sums of Fourier coefficients over intervals.[17] Due to the $O(1)$ gap in value between the two cases, distinguishing can achieved in constant time and query complexity.

Consider next the general case when there's no promise on the location of the heaviest Fourier coefficient. In this case, we partition[18] the interval $[-\frac{N}{4}, \frac{N}{4}]$ to $O(1)$ intervals I that are symmetric, sufficiently small and bounded away from $\pm\frac{N}{4}$; apply the above distinguishing procedure on each of those intervals; and output 1 iff at least one of these procedures outputted YES indicating that the heaviest Fourier coefficient is in I.

We sketch that analysis of this algorithm. We consider separately the three following cases regarding the location of the heaviest Fourier coefficient of C'_{ms}: (i) The heaviest coefficient is in one of the intervals I in the partition, (ii) The heaviest coefficient is not in $[-\frac{N}{4}, \frac{N}{4}]$ and moreover, it is bounded away from $\pm\frac{N}{4}$, and (iii) The

[14]We say that an interval $I \subseteq \mathbb{Z}_N$ is *symmetric* if there are integers $0 \le a < b < N$ s.t. $I = [a, b] \bigcup [-b, -a]$ where $[a, b] = \{\alpha \in \mathbb{Z}_N \mid a \le \alpha \le b\}$ and $[-b, -a] = \{\alpha \in \mathbb{Z}_N \mid N - b \le \alpha \le N - a\}$.

[15]For an interval $I \subseteq \mathbb{Z}_N$ and an element $a \in \mathbb{Z}_N$, we say that I is *bounded away* from $a \in \mathbb{Z}_N$ if I does not contain any elements that are close to a, that is, $I \bigcap [a(1 - O(1)), a(1 + O(1))] = \phi$.

[16]To prove this reduction we analyze the Fourier spectrum of C_{ms} showing that its heaviest Fourier coefficient α_{max} is of squared magnitude at least $|C_{ms}(\alpha_{max})|^2 > \frac{1}{4} + O(1)$, and moreover, the Fourier spectrum is symmetric, that is, $|C_{ms}(\alpha)|^2 = |C_{ms}(N - \alpha)|^2$ for all $\alpha \in \mathbb{Z}_N$. This implies that if the character α_{max} with heaviest Fourier coefficient lies in I, then so does $N - \alpha_{max}$ (due to symmetry of I), and thus the sum of squared Fourier coefficients over I is greater than twice the maximal Fourier coefficient, that is, $\frac{1}{2} + O(1)$. Moreover, $weight(I) > \frac{1}{2} + O(1)$ implies that $weight(I') < \frac{1}{2} - O(1)$ for any disjoint interval I', because by Parseval identity for Boolean functions the sum of all squared Fourier coefficients is at most 1. The same holds for the corrupted codewords C'_{ms} with the considered noise bound.

[17]Specifically, to approximate the Fourier weight of an interval I we estimate the norm of the convolution of C'_{ms} with a filter that attenuates coefficients outside of I. Estimating this norm is by taking appropriate averages over random (though correlated) entries of C'_{ms}.

[18]More precisely, the intervals I form a partition of $[-(1 - O(1))\frac{N}{4}, (1 - O(1))\frac{N}{4}]$, as they are all bounded away from $\pm\frac{N}{4}$.

heaviest coefficient is not bounded away from $\pm\frac{N}{4}$. In the first case, the interval I, in which the heaviest Fourier coefficient lies, satisfies the conditions of the special case considered above, and thus the algorithm outputs 1 (with high probability). In the second case, each interval I in the partition satisfies the conditions of special case considered above, and thus all runs return NO, and the algorithm returns 0 (with high probability). In the third case, we do not have a guarantee regarding the output of the algorithm, and the algorithm may return a false answer. Nevertheless, the probability that this third case occurs is small: The heaviest Fourier coefficient is not bounded away from $\pm\frac{N}{4}$ iff ms is in a small neighborhood around $\frac{N}{4}$ or $-\frac{N}{4}$. The probability that this event happens for a random entry s is proportional to the size of this neighborhood, because the map from entry s to location ms of the heaviest Fourier coefficient of C_{ms} is one-to-one.

6.2.3 Distance Bounding using Fourier Analysis

We bound the distance of the codes $\mathcal{C}^{\mathcal{P}} = (\mathbb{Z}_N, P_N, \mathbb{Z}_N)$ for groups $\mathbb{Z}_N$. We lower bound the distance of these codes in terms of the Fourier spectrum of the alphabet controlling functions $P_N\colon \mathbb{Z}_N \to \mathbb{C}$.[19] To enable giving our distance bound, we slightly restrict the message space of the considered codes to be the set

$$M_N = \mathbb{Z}_N^* \cap \left\{1, \ldots, \lfloor(\frac{N}{2})\rfloor\right\}$$

This restriction incurs only a small reduction in size of the message space.

Theorem 6.27. *For any $N \in \mathbb{N}$ and $P_N\colon \mathbb{Z}_N \to \{\pm 1\}$, the Multiplication code $(\mathbb{Z}_N, P_N, \mathbb{Z}_N)$ with message space M_N is a code of rate $(\log N - O(1))/N$ and distance at least*

$$\left(\left|\widehat{P_N}(\alpha_1)\right| - \left|\widehat{P_N}(\alpha_2)\right|\right)^2$$

for $\alpha_1, \alpha_2 \in \mathbb{Z}_N$ defined by

$$\alpha_1 = \arg\max_{\alpha \in \mathbb{Z}_N} \left|\widehat{P_N}(\alpha)\right|$$

$$\alpha_2 = \arg\max_{\alpha \in \mathbb{Z}_N, \alpha \neq \pm\alpha_1} \left|\widehat{P_N}(\alpha)\right|$$

where $\widehat{P_N}(\alpha) = \mathbb{E}_{x \in \mathbb{Z}_N}\left[P_N(x) e^{i\frac{2\pi}{N}\alpha x}\right]$.

In particular, we show the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ have constant distance. To show this we analyze the Fourier spectrum of the functions half_N bounding from

[19] To relate the distance of two codewords C_1, C_2 to their Fourier spectrum we first shift to ± 1 notation of binary values, then express the distance in terms of inner product: $\Delta(C_1, C_2) = \frac{1}{2} - \langle C_1, C_2 \rangle$, and finally observe that by Plancharel Identity $\langle C_1, C_2 \rangle = \langle \widehat{C}_1, \widehat{C}_2 \rangle$. To bound this expression we then employ the definition of codewords C_1, C_2 of $\mathcal{C}^{\mathcal{P}}$ to show that their two heaviest Fourier coefficients cannot collide. This in turn implies a bound on the inner product of their Fourier spectrums $\langle \widehat{C}_1, \widehat{C}_2 \rangle$.

below their heaviest Fourier coefficient by $\frac{2}{\pi}$, and bounding from above their second heaviest Fourier coefficient by $\frac{2}{3\pi}\left(1+O\left(\frac{1}{N}\right)\right)$.

Corollary 6.28. *For any $N \in \mathbb{N}$, the Multiplication code $(\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ with message space M_N is a binary code of rate $(\log N - O(1))/N$ and distance at least $\left(\frac{4}{3\pi}\right)^2\left(1-O\left(\frac{1}{N}\right)\right) \approx 0.18$.*

6.2.4 Codes for Groups with Small Generating Set

We present MPC codes for groups G of small generating sets.

Our codes for groups of small generating sets are constructed from our codes for the groups $\mathbb{Z}_{N_i}$, for N_i the orders of the generators of G:

- When $G = \mathbb{Z}_N^k$, encoding a message $m = (m_1, \dots, m_k) \in \mathbb{Z}_N^k$ is done by first encoding each coordinate $m_i \in \mathbb{Z}_N$ by a code for the group $\mathbb{Z}_N$ and then combining the resulting k codewords into one codeword in which each symbol is the concatenation of the k corresponding symbols.

- When $G = \mathbb{Z}_{N_1} \times \dots \times \mathbb{Z}_{N_k}$ has generators of varying orders, encoding is again done by first encoding each coordinate $m_i \in \mathbb{Z}_{N_i}$ by a code for $\mathbb{Z}_{N_i}$ and then combining all these codewords together into one codeword. A combining step different than the one used for $\mathbb{Z}_N^k$ is however needed due to a discrepancy in length between the resulting codewords. The combining step we use relies on an expander graph to ensure a good mixing of all codewords symbols.

- When G is an arbitrary finite abelian group with generators of orders $N_1, \dots, N_k$ we exploit the isomorphism between G and a group $G' = \mathbb{Z}_{N_1} \times \dots \times \mathbb{Z}_{N_k}$ to derive codes for G from the codes for G'.

We elaborate on each of these cases below.

Codes for $\mathbb{Z}_N^k$.

We present Multiplication codes $\mathcal{C}_{\mathbb{Z}_N^k} = (\mathbb{Z}_N^k, P_{\mathbb{Z}_N^k}, S_{\mathbb{Z}_N^k})$ for groups $\mathbb{Z}_N^k$. We construct these codes using as a building-block codes $\mathcal{C}_{\mathbb{Z}_N} = (\mathbb{Z}_N, P_N, S_N)$ for the groups $\mathbb{Z}_N$. In the codes $\mathcal{C}_{\mathbb{Z}_N^k}$, encoding a message $m = (m_1, \dots, m_k) \in \mathbb{Z}_N^k$ is done as follows. First, each coordinate $m_i \in \mathbb{Z}_N$ is encoded by the code $\mathcal{C}_{\mathbb{Z}_N}$. This results in k codewords of length $|S_N|$. Next, these k codewords are joint together into one codeword of length $|S_N|$ in which each symbol is the concatenation of the k corresponding symbols.

Definition 6.29. *Given a Multiplication code $\mathcal{C}_{\mathbb{Z}_N} = (\mathbb{Z}_N, P_N, S_N)$ for the group $\mathbb{Z}_N$ with alphabet controlling function $P_N \colon \mathbb{Z}_N \to \Sigma$, we define a Multiplication code $\mathcal{C}_{\mathbb{Z}_N^k} = (\mathbb{Z}_N^k, P_{\mathbb{Z}_N^k}, S_{\mathbb{Z}_N^k})$ for the group $\mathbb{Z}_N^k$ as follows. The alphabet controlling function $P_{\mathbb{Z}_N^k} \colon \mathbb{Z}_N^k \to \Sigma^k$ is defined by*

$$P_{\mathbb{Z}_N^k}(y_1, \dots, y_k) = (P_N(y_1), \dots, P_N(y_k))$$

and the set of indexes $S_{\mathbb{Z}_N^k} \subseteq \mathbb{Z}_N^k$ is defined by

$$S_{\mathbb{Z}_N^k} = \left\{s^k\right\}_{s\in S_N}$$

where $s^k \in \mathbb{Z}_N^k$ is the elements with all k coordinates equal to s.

Properties of the resulting code $\mathcal{C}_{\mathbb{Z}_N^k} = (\mathbb{Z}_N^k, P_{\mathbb{Z}_N^k}, S_{\mathbb{Z}_N^k})$ naturally depend on properties of the building-block codes $\mathcal{C}_{\mathbb{Z}_N} = (\mathbb{Z}_N, P_N, S_N)$. In terms of combinatorial properties: $\mathcal{C}_{\mathbb{Z}_N^k}$ has alphabet size 2^k, and rate and distance equal to those of $\mathcal{C}_{\mathbb{Z}_N}$.[20] In terms of algorithmic properties, the derivation of properties of $\mathcal{C}_{\mathbb{Z}_N^k}$ from properties of $\mathcal{C}_{\mathbb{Z}_N}$ is as follows. If $\mathcal{C}_{\mathbb{Z}_N}$ is $(\varepsilon, \ell, q, T)$-list decodable then $\mathcal{C}_{\mathbb{Z}_N^k}$ is $(\varepsilon, \ell^k, q, kT)$-list decodable; where we say that a code is $(\varepsilon, \ell, q, T)$-list decodable if there is an algorithm that given a corrupted codeword w and a distance parameter ε output a list of length at most ℓ containing all codewords at distance at most ε from w in running time time T and with q queries to w. If $\mathcal{C}_{\mathbb{Z}_N}$ is $(\varepsilon, q, T, 1-\rho)$-locally self correctable then $\mathcal{C}_{\mathbb{Z}_N^k}$ is $(\varepsilon, q, kT, (1-\rho)^k)$-locally self correctable.

Theorem 6.30. *Let $\mathcal{C}_{\mathbb{Z}_N}$ be a code for the group $\mathbb{Z}_N$ of alphabet Σ, rate r, relative Hamming distance Δ, which is $(\varepsilon, \ell, q, T)$-list decodable and $(\varepsilon, q, T, 1-\rho)$-locally self correctable. Let $\mathcal{C}_{\mathbb{Z}_N^k}$ be the code for the group $\mathbb{Z}_N^k$ built from the code $\mathcal{C}_{\mathbb{Z}_N}$ as in Definition 6.29. Then the code $\mathcal{C}_{\mathbb{Z}_N^k}$ is a code for the group $\mathbb{Z}_N^k$ of alphabet Σ^k, rate r, relative Hamming distance Δ, which is $(\varepsilon, \ell, q, kT)$-list decodable and $(\varepsilon, q, kT, (1-\rho)^k)$-locally self correctable.*

To prove the above algorithmic properties we assume there exists an algorithm A for $\mathcal{C}_{\mathbb{Z}_N}$, and use it to construct an algorithm A' for $\mathcal{C}_{\mathbb{Z}_N^k}$. The algorithm A' operates as follows: First, A' decompose the given corrupted codeword into the k corrupted codeword corresponding to the codewords in $\mathbb{Z}_N$ encoding the symbols $m_1, \ldots, m_k$ of the message $m = (m_1, \ldots, m_k) \in \mathbb{Z}_N^k$. Second, A' runs the algorithm A on each corrupted codeword. Finally A' combines the outputs of all k runs of the algorithm A to one output of the algorithm A'. The combining step differs depending on the task at hand. For (local) list decoding, the output of the i-th run of A is a list $L_i \subseteq \mathbb{Z}_N$ of length ℓ, and A' outputs the product list $L = \{(x_1, \ldots, x_k) \mid x_i \in L_i, i = 1, \ldots, k\}$ of length ℓ^k. For unique decoding, the output of the i-th run of A is a single value $m'_i \in \mathbb{Z}_N$ corresponding to the i-th symbol of the encoded message m, and A' outputs the product of all these values $m' = (m'_1, \ldots, m'_k)$. For local self correcting, the output of the i-th run of A is a single bit b_i indicating the value of codeword encoding each m_i on entry s, and A' outputs the product of all these values $v = (b_1, \ldots, b_k) \in \{0,1\}^k$.

[20]In details, the rate of $\mathcal{C}_{\mathbb{Z}_N^k}$ is $\frac{\log_{2^k}\left|(\mathbb{Z}_N^k)^*\right|}{\left|S_{\mathbb{Z}_N^k}\right|}$ (for $(\mathbb{Z}_N^k)^*$ the set of invertible elements in $\mathbb{Z}_N^k G$), which is equal to the rate $\frac{\log_2|\mathbb{Z}^*|}{\left|S_{\mathbb{Z}_N}\right|}$ of $\mathcal{C}_{\mathbb{Z}_N}$, because $\log_{2^k}\left|(\mathbb{Z}_N^k)^*\right| = \frac{1}{k}\log_2\left|(\mathbb{Z}_N^k)^*\right|$ and $\left|(\mathbb{Z}_N^k)^*\right| = \left|\mathbb{Z}_N^*\right|^k$. The distance of $\mathcal{C}_{\mathbb{Z}_N^k}$ is clearly at least as large as the distance of $\mathcal{C}_{\mathbb{Z}_N}$, because restricting the blocks of each symbol to, say, the first bit we get the distance of $\mathcal{C}_{\mathbb{Z}_N}$. The distance of $\mathcal{C}_{\mathbb{Z}_N^k}$ is at most the distance of $\mathcal{C}_{\mathbb{Z}_N}$, because for each $m, m' \in \mathbb{Z}_N$ the distance of the encodings of m^k and m'^k in $\mathcal{C}_{\mathbb{Z}_N^k}$ are equal to the distance of the encodings of m and m' in $\mathcal{C}_{\mathbb{Z}_N}$.

Combining the derivation rules in Theorem 6.30 with the properties of our Multiplication codes for $\mathbb{Z}_N$ proves there exists codes for the group $\mathbb{Z}_N^k$ achieving parameters as stated in Theorem 6.6

Codes for $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$.

We present Multiplication codes for groups $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$. We construct these codes using as a building-block codes $(\mathbb{Z}_{N_i}, P_{N_i}, S_{N_i})$ for the groups $\mathbb{Z}_{N_i}$, $i \in [k]$.

We describe the algorithm for encoding messages $m = (m_1, \ldots, m_k) \in G$. In a high level, the encoding is similar to the one for $\mathbb{Z}_N^k$, namely, we first encode each coordinate $m_i \in \mathbb{Z}_{N_i}$ by the code $(\mathbb{Z}_{N_i}, P_{N_i}, S_{N_i})$ for the group $\mathbb{Z}_{N_i}$, and then combine all these codewords together into one codeword. The combining step however is different from the one used for $\mathbb{Z}_N^k$, due to the discrepancy in length between the codewords C_{m_i}. Let us first demonstrate why the combining algorithm used for $\mathbb{Z}_N^k$ is not satisfactory for the case $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, and then describe the combining step we do use here.

To demonstrate why the combining algorithm used for $\mathbb{Z}_N^k$ won't do, consider for example encoding a message $m = (m_1, m_2)$ in the group $\mathbb{Z}_2 \times \mathbb{Z}_N$ using the code $(\mathbb{Z}_2, \mathsf{half}_2, \mathbb{Z}_2)$ to encoding m_1 and the code $(\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ to encode m_2. The resulting codewords have lengths 2 and N respectively. Using the former combining algorithm results in a codeword of length N where the first two symbols are a concatenation of values from each of the two codewords, whereas the other symbols are simply the values of the second codeword. This results in a code of a very bad distance: by flipping only the first two symbols of the resulting codeword all information regarding m_1 is lost and no decoding is possible.

Our combining step for $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ mixes together the entries of all codewords in a way ensuring that to destroy information on any m_i an adversary must destroy a considerable fraction of the symbols in the combined codeword. To achieve this guarantee we rely on properties of expander graphs as follows. Denote $n = \max_{i \in [k]} |S_{N_i}|$ and let $H_i = ([N_i], [n], [E_i])$ be a bipartite expander graph of right degreed $d = O(1)$, $\forall i \in [k]$. To combine all codewords we first encode each C_{m_i} using ABNNR encoding [7] corresponding to the graph H_i. That is, we encode C_{m_i} to a length n codeword $C_{m_i}^{H_i}$ with alphabet $\{0,1\}^d$ s.t. its j-th entry is the concatenation of the values of C_{m_i} on entries $j_1, \ldots, j_d$ the neighbor of right node j in the graph H_i. This results in k codewords $C_{m_1}^{H_1}, \ldots, C_{m_k}^{H_k}$ each of length n and alphabet $\{0,1\}^d$. We now combine these codewords by defining a new codeword C_m whose j-th entry is the concatenation of the j-th entries of all codewords $C_{m_1}^{H_1}, \ldots, C_{m_k}^{H_k}$. This results in a codeword of length n and alphabet $\{0,1\}^{kd}$. The set of indexes to the entries of this codewords is $S_G \subseteq G^d$ the set of all elements $((s_1^{1j}, \ldots, s_k^{1j}), \ldots, (s_1^{dj}, \ldots, s_k^{dj})) \in G^d$ with $s_i^{\ell j}$ the ℓ-th neighbor of right node j in expander graph H_i.

Definition 6.31. *Given binary Multiplication codes $(\mathbb{Z}_{N_t}, P_{N_t}, S_{N_t})$ for $t = 1, \ldots, k$ and bipartite expander graphs $H_t = ([N_t], [n], [E_t])$ of right degreed $d = O(1)$, for each $t = 1, \ldots, k$, we define a Multiplication code (G, P_G, S_G) for the group $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ as follows. The alphabet controlling function $P_G \colon G^d \to \{0,1\}^{kd}$ is* *defined by*

$$P_G((y_1^1,\ldots,y_k^1),\ldots,(y_1^d,\ldots,y_k^d)) = ((P_{N_1}(y_1^1),\ldots,P_{N_k}(y_k^1)),\ldots,(P_{N_1}(y_1^d),\ldots,P_{N_k}(y_k^d)))$$

and the set of indexes $S_G \subseteq G^d$ is defined by

$$S_G = \left\{((s_1^{1j},\ldots,s_k^{1j}),\ldots,(s_1^{dj},\ldots,s_k^{dj}))\right\}_{j\in[N_k]}$$

for s_t^{ij} the i-th neighbor of right node j in the graph H_t.

The combinatorial and algorithmic properties of the resulting code $\mathcal{C}_G = (G, P_G, S_G)$ naturally depend on properties of the building-block codes $\mathcal{C}_{N_i} = (\mathbb{Z}_{N_i}, P_{N_i}, S_{N_i})$. In terms of combinatorial properties: $\mathcal{C}_G$ has alphabet size 2^{dk}, rate[21] $O(\frac{1}{kd}\frac{\log N}{N})$ for $N = \max_{i\in[k]} N_i$, and distance $1 - O_d(1)$.[22] In terms of algorithmic properties, if all codes $\mathcal{C}_{N_i}$ are uniquely decodable, then so is the code $\mathcal{C}_G$; likewise, if all codes $\mathcal{C}_{N_i}$ are self correctable, then so is the code $\mathcal{C}_G$. The parameters of the algorithms for $\mathcal{C}_G$ are derived from those for $\mathcal{C}_{N_i}$ as described in the theorem below.

We remark that the current encoding does not admit to list decoding. The bottle neck is the ABBNR encoding layer. Possibly the ABNNR codes could be substituted with the list decodable code of Guruswami and Indyk [46] to achieve list decodable codes for G.

Theorem 6.32. *For $i = 1,\ldots,k$, let $\mathcal{C}_{\mathbb{Z}_{N_i}}$ be codes for the groups $\mathbb{Z}_{N_i}$ of alphabet Σ, rate r, relative Hamming distance Δ, which are $(O(1),\ell,q,T)$-decodable and $(O(1),q,T,1-\rho)$-locally self correctable. Let $\mathcal{C}_G$ be the code for the group $G = \mathbb{Z}_{N_1}\times\ldots\times\mathbb{Z}_{N_k}$ built from the code $\mathcal{C}_{\mathbb{Z}_N}$ as in Definition 6.31. Then the code $\mathcal{C}_G$ is a code for the group $\mathbb{Z}_N^k$ of alphabet Σ^k, rate r, relative Hamming distance $1-O(1)$, which is $(\frac{1}{2}-O(1),1,q,kT)$-decodable and $(\frac{1}{2}-O(1),q,kT,(1-\rho)^k)$-locally self correctable.*

To prove the above algorithmic properties we construct an algorithm A' for our codes for G from given algorithms $A_1,\ldots,A_k$ for the codes for $\mathbb{Z}_{N_1},\ldots,\mathbb{Z}_{N_k}$. The first step in the algorithm A is a decomposing step where the corrupted codeword of $\mathcal{C}_G$ is decomposed to k corrupted codewords of $\mathcal{C}_{\mathbb{Z}_{N_1}},\ldots,\mathcal{C}_{\mathbb{Z}_{N_k}}$. This decomposition step is done by tracing back the combining step in the encoding and decoding the ABNNR encoding layer. A subtle point is that the decomposition step requires *local decoding.*[23] That is, the entries of the k codewords are produced in time independent of the length of the given corrupted codeword. This is necessary for example, when the encoding length is exponential, or for the local self correcting algorithm. We

[21]The rate is $\frac{1}{dk}\frac{O(\log N)}{N} = O_{dk}(\frac{\log N}{N})$, because the encoding length is $N = \max_{i\in[k]} N_i$ and the information rate is $\log_{2^{kd}}|G^*| \geq \frac{1}{kd}O(\log N)$.

[22]The distance is at least the distance of the code when restricting each block to the values of $\mathcal{C}_{m_i}^{H_i}$ for some fixed $i\in[k]$. The distance of this restriction is the distance of the ABNNR code which is $1-O_d(1)$ as shown by [7].

[23]More accurately, the decomposition procedure is a local *error reduction* procedure, namely, given an entry location in one of the k codewords, it output the correct value on this location with high probability. Thus the decomposition procedure provides access to k *corrupted* codewords.

achieve local decoding with a constant number of queries –even in cases when the degree of the expander graph is non constant– by applying the decoding algorithm on a constant size neighbors set chosen by random sampling. On the chosen set of neighbors, decoding may be done using the majority vote rule as in Guruswami-Indyk [44].The rest of the steps of algorithm A' are constructed from $A_1, \ldots, A_t$ similarly to the our construction of algorithms for $\mathbb{Z}_N^k$ from algorithms for $\mathbb{Z}_N$.

The derivation rules of Theorem 6.32 combined with the properties of our Multiplication codes for $\mathbb{Z}_N$ yield codes achieving parameters as stated in Theorem 6.6.

Codes for Groups with Small Generating Set

For G an arbitrary finite abelian group, there is an isomorphism between G and a group $G' = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ for $N_1, \ldots, N_k$ the orders of generators in a generating set of G. We exploit this isomorphism to derive Multiplication codes for the group G from our Multiplication codes for the group G'.

Definition 6.33. *Let G be a group generated by $g_1, \ldots, g_k$ of orders $N_1, \ldots, N_k$, denote by G' the direct product group $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ and let $\mathcal{C}_{G'} = (G', P_{G'}, S_{G'})$ be a Multiplication code for the group G'. We define a Multiplication code $\mathcal{C}_G = (G, P_G, S_G)$ for the group G by setting*

$$P_G(\prod_{i=1}^{k} g_i^{x_i}) = P_{G'}(x_1, \ldots, x_k)$$

and

$$S_G = \left\{ (\prod_{i=1}^{k} g_i^{x_{i1}}, \ldots, \prod_{i=1}^{k} g_i^{x_{id}}) \in G^d \;\middle|\; ((x_{11}, \ldots, x_{k1}), \ldots, (x_{1d}, \ldots, x_{kd}) \in G'^d \right\}$$

The resulting Multiplication for G' code achieve alphabet, rate, distance and decodability parameters identical to those of the Multiplication codes for G. Combined with our results on codes for $\mathbb{Z}_N$, this completes the proof of Theorem 6.6

6.2.5 Linear Encoding Length via Restrictions

We present codes for finite abelian groups of linear encoding length, constant distance and constant list decoding bound. We do not have however an efficient decoding algorithm for this code.

Theorem 6.34 (Linear encoding length)**.** *For any $\gamma < 0.17$ and any finite abelian group G, there exists Multiplication codes $\mathcal{C}$ for G of alphabet Σ, rate $O(\frac{1}{\gamma^2 \log|\Sigma|})$, constant distance, and satisfying the following list decoding bound: $\left|Ball_{\mathcal{C}}(C, \frac{1}{2} - 2\gamma)\right| \leq poly(\log|G|, 1/\gamma)$, $\forall C \in \mathcal{C}$.*[24]*. The alphabet size $|\Sigma|$ depends on the underlying group: $|\Sigma| = 2$ for cyclic groups, $|\Sigma| = O(1)$ for groups with constant size generating set, and*

[24]$Ball_{\mathcal{C}}(C, r)$ denotes the set of codewords of $\mathcal{C}$ within relative Hamming distance at most r from C.

$|\Sigma| = |G|$ for arbitrary finite abelian groups. Algebraic operations in these decoding algorithms are computed in the group G.

The codes of linear encoding length are constructed by taking *random restrictions* of the previously constructed codes achieving exponential encoding, constant distance and constant list decoding bound. The restriction $\mathcal{C}(S')$ of a code $\mathcal{C} = (G, P, S)$ to a subset $S' \subseteq S$ of its codewords entries is the code whose codewords are restrictions of the codewords of $\mathcal{C}$ to the entries indexed by elements in S'. We say that the code $\mathcal{C}(S')$ is an *n-random restriction*, if S' is a set of size n chosen uniformly at random from S. We show that the code $\mathcal{C}(S')$ maintains roughly the same distance and list decoding bound as $\mathcal{C}$, while achieving linear encoding length. We thus obtain codes of linear encoding length and constant distance and list decoding bound.

Definition 6.35 (Restrictions)**.** *For any code $\mathcal{C} \subseteq \Sigma^S$ with entries indexed by elements in S and a subset $S' \subseteq S$, we defined the* restriction of $\mathcal{C}$ to S' *to be the code $\mathcal{C}(S') = \left\{C_{|S}\right\}_{C\in\mathcal{C}}$ for $C_{|S} \in \Sigma^S$ the restriction of C to entries whose index s is in S'. We say that $\mathcal{C}(S')$ is an n-*random restriction of $\mathcal{C}$ *if the set S' is a a subset of S of size n chosen uniformly at random.*

Lemma 6.36. *Let $\mathcal{C} \subseteq \Sigma^S$ be a code of non-linear encoding length $|S| = \omega(\log|\mathcal{C}|)$ and with relative Hamming distance at least δ. For any $\gamma > 0$ and any $n = \Theta(\log N/\gamma^2)$ sufficiently large, the n-random restriction of $\mathcal{C}$ has linear encoding length of $O(1/\gamma^2)$ and its relative Hamming distance is at least $\delta - \gamma$ with probability at least $1 - O(\frac{1}{|\mathcal{C}|})$ (where the probability is taken over the choice of the random restriction $S' \subseteq S$).*

The problem of decoding these codes is related to the problem of Learning Characters with Noise in the *random samples access model* (aka, rLCN); see Definition 4.1. Specifically, an efficient solution to rLCN would imply an efficient decoding algorithm for these constant rate codes. We conjecture however that rLCN is intractable.

We use the following terminology and notations for defining the relation between the algorithmic properties of $\mathcal{C}^{const}$ and rLCN: We say that algorithm A *(s,t,e)-solves rLCN*, if, on input a description of a finite abelian group G, a threshold γ and random samples access to $f\colon \mathbb{Z}_N \to \mathbb{C}$, A asks for s samples of f, runs in time t, and outputs $L \supseteq \mathsf{Heavy}_\gamma(f)$ with probability at least $1 - e$.

For each γ, we denote by $\varepsilon(\gamma)$ the threshold guaranteed by the Agreement lemma 6.46 for which $\Delta(w, C_{N,x}) < \varepsilon$ implies that $\exists 0 \neq \beta \in \mathsf{Heavy}_\gamma(w) \cap \mathsf{Heavy}_\gamma(C_{N,x})$. In particular, for $\mathcal{P} = \Psi$, $\varepsilon(\gamma) = O(\sqrt{\gamma})$.

Theorem 6.37 (Algorithmic properties)**.** *For any group G, $\gamma > 0$ and $\mathcal{C}$ a concentrated and recoverable code, if there is an algorithm A that (s,t,e)-solves rLCN with $s = O(\frac{\log|G|}{\gamma^2})$, $t = poly(\log|G|, \frac{1}{\gamma})$ and $e = poly(1/|G|)$, then the $O(\log|G|/\gamma^2)$-random restriction code of $\mathcal{C}$ is algorithmically $(\frac{1}{2} - \gamma - \varepsilon, poly(\log|G|, 1/\varepsilon))$-list decodable, with probability at least $2/3$ (where the probability is taken over the choice of random restriction).*

We remark that the random restriction codes can be viewed as a non-linear analogue of random linear codes. The analogy between these codes and random linear codes is both in the definition of the codes, where we observe that random linear codes can be viewed as restrictions of the Hadamard codes to random subsets of the Boolean Cube $\mathbb{F}_2^k$. The analogy is also the codes properties, where both the constant rate Multiplication Codes and the random linear codes exhibit *good combinatorial properties*, but *poor algorithmic properties.*

6.2.6 Soft Local Error Reduction for Concatenated ABNNR Codes

In this section we present our soft error reduction algorithm for concatenated ABNNR codes.

Alon *et.al.* [7] presented a distance amplification scheme for error correcting codes relying on expansion properties of expander graphs. Their scheme produces from a base code C_0 of alphabet q and distance $\delta = O(1)$ a new code –aka, *ABNNR code*– of alphabet $O(q)$ and distance $1 - O(1)$. To achieve a *binary* code, one can *concatenate* the ABNNR code with a small binary code, that is, encode each symbol in the codeword of the (non-binary) ABNNR code with a binary code. Following standard terminology, we refer to the non-binary code as the *inner code* and to the binary code concatenated with it as the *outer code.*

Guruswami and Indyk [44] presented a decoding algorithm for codes concatenated with the ABNNR codes. The decoding algorithm of [44] follows Forney's GMD methodology [33] for decoding concatenated codes requiring an efficient error reduction algorithm for the outer code, and an algorithm for decoding from erasures and errors for the inner code.

We present an alternative to the Forney decoding approach that does not require an erasures-and-errors decoding algorithm for the inner code. Instead we take a soft decoding approach where the outer code returns a list of potential codewords (of the outer code) along with their distances from the received word, and the inner code incorporates these lists of distances to find the closest codeword.

The algorithm we present is *local*, namely, each bit of message symbol can be found with a d^2 queries to the codewords for d the degree of the underlying expander graph.

Our soft decoding approach for decoding concatenated ABNNR codes gives the same optimal distance performance as the Forney decoding approach employed in [44], and may be of interest outside the realm of Multiplication codes, *e.g.*, for decoding concatenated codes where the inner code has no efficient erasures-and-errors decoding algorithm.

In our context, we use the soft decoding approach in decoding our *binary* codes for groups of few generators of varying orders, which were obtained by concatenating the ABNNR layer with a small binary code (see Remark 6.7).

To state our theorem and present our soft error reduction algorithm, let us first present the definitions of expander graphs and ABNNR codes concatenated with binary codes.

Definition 6.38 (Expander Graphs)**.** *Let $H = ([n],[n],E)$ be a d-regular bipartite graph. Let λ be the second largest (in absolute value) eigenvalue of the normalized*

Figure 6-1: Illustration of a binary ABNNR code

adjacency matrix of H, then, we say that H is a λ-expander. *In particular, When $\lambda \leq 2/\sqrt{d}$, we say that H is a* Ramanujan graph.

Fact 6.39. *For any $d \geq 3$, and for all $n \in \mathbb{N}$, a random d-regular bipartite graph $H = ([n],[n],E)$ is Ramanujan with high probability.*

Definition 6.40 (binary ABNNR code [7]). *Let $\mathcal{H} = \{([n],[n],E_n)\}_{n\in\mathbb{N}}$ be a family of d-regular λ-expander graphs, and let $\mathcal{C}_0$ be a family of binary codes encoding d bits into $O(d)$ bits. Let $m = m_1 \ldots m_n$ be a message in $\{0,1\}^n$, we define the encoding of m in the ABNNR and binary ABNNR codes (Illustration provided in Figure 6-1):*

1. **ABNNR code.** *The ABNNR code $\mathcal{C}_{\mathcal{H}}$ encodes m by the codeword $C_{H_n}(m) \in (\{0,1\}^d)^n$ whose i-th symbol is $(m_{i_1},\ldots,m_{i_d})$ for $i_1,\ldots,i_d$ the left neighbors of right node i in the graph H_n.*[25]

2. **binary ABNNR code.** *Denote by $\mathcal{C}(\mathcal{H},\mathcal{C}_0)$ the ABNNR code $\mathcal{C}_{\mathcal{H}}$ concatenated with $\mathcal{C}_0$. That is, the codeword in $\mathcal{C}(\mathcal{H},\mathcal{C}_0)$ encoding message m is the vector $C(m) = (\mathcal{C}_0(C_{H_n}(m)_1),\ldots,\mathcal{C}_0(C_{H_n}(m)_n) \in (\{0,1\}^{O(d)})^n$.*

   Notations. *We use the terminology "*the i-th block of $C(m)$*" to refer to bits $\mathcal{C}_0(C_{H_n}(m)_i)$ of $C(m)$. We denote the restriction of $C(m)$ to its i-th block by $C(m)^i$. Likewise, for a corrupted codeword w, we denote by w^i the restriction of w to the bits corresponding to the i-th codeword block.*

Remark 6.41. *ABNNR code is often employed for encoding messages m which are in themselves codewords of some code $\mathcal{C}_1$ of (small) constant distance and constant rate. This is because in order for the ABNNR codes to* amplifies *distance, there must be some initial distance to begin with. See illustration in Figure 6-1.*

We present a soft error reduction algorithm that, given a corrupted codeword w of a binary ABNNR code, recovers $(1-O(1))$-fraction of the bits of the encoded message x. The soft error reduction algorithm works by finding independently for each left node ℓ the best assignment to its neighbors $\Gamma(\ell)$, as follows. For each left node ℓ, we restrict our attention to the corrupted codeword block w^i corresponding to right neighbors $i \in \Gamma(\ell)$ of ℓ. We then find the codeword $C(m')$ closest to w^i on those blocks (by exhaustive search). If the closest codeword is sufficiently close, we set the ℓ-th output bit z_ℓ to be the value of the ℓ-th bit of the encoded message m'. We repeat this procedure for each left node $\ell \in [n]$, outputting $z_1,\ldots,z_n$. A pseudocode follows.

Algorithm 6.42. Soft Error Reduction Algorithm.
Input: *A description of the code $\mathcal{C}(\mathcal{H},\mathcal{C}_0)$, a noise parameter ε, and a corrupted codeword $w \in \{0,1\}^{n/r}$ s.t. $\Delta(C(x),w) \leq \frac{1}{4} - \frac{\varepsilon}{4}$.*

[25]The ABNNR code applies to any alphabet Σ. We focus on binary alphabet for simplicity.

Output: $z \in \{0,1\}^n$ *s.t.* $\Delta(z,x) \le \lambda^{1/3}$.
Steps:

1. *For each left node* $\ell \in [n]$
   (a) *By exhaustive search, find assignment* $y \in \{0,1\}^{d^2}$ *to all message bits* $i \in \Gamma_L(\Gamma_R(\ell))$ *such that* y *minimizes*
   $$dist_\ell(y) = \frac{1}{d}\sum_{i\in\Gamma(\ell)} \Delta(C_0(y)^i, w^i)$$
   (b) *If* $dist_\ell(y) < \frac{1}{4} - \frac{\varepsilon}{4}$, *set* $z_\ell = y_\ell$
   (c) *Else set* $z_\ell = \perp$
2. *Output* $z_1 \dots z_k$

To show that this is an efficient algorithm, we argue that the exhaustive search step can be done in constant running time. This is because there is a $d = O(1)$ number of blocks i neighboring each left node ℓ, and each of these blocks is determined by a $d = O(1)$ number of message bits (specifically, the bits corresponding to left neighbors of right node i). Thus, the number of assignment to be examined in the exhaustive search is $2^{d^2} = O(1)$.

To prove the correctness of this algorithm, we first apply the Expander Mixing Lemma to show that for the correct message x, $dist_\ell(x) < \frac{1}{4} - \frac{\varepsilon}{4}$ for at least $(1 - O(\lambda^{1/3}))$-fraction of the bits $\ell \in [n]$. Second, we show that for any y that disagrees with x on the ℓ-th bit (*i.e.*, $y_\ell \neq x_\ell$), $dist_\ell(y) > \frac{1}{4} - \frac{\varepsilon}{4}$. Combined together this proves that the output z of the algorithm agrees with the encoded message x on at least $(1 - O(\lambda^{1/3}))n$ of its bits.

Theorem 6.43 (Soft Error Reduction)**.** *Let $\mathcal{C}(\mathcal{H}, \mathcal{C}_0)$ be a binary ABNNR code with $\mathcal{H}$ a family of d-regular λ-expanders, and $\mathcal{C}_0$ a family of binary error correcting codes of rate r and relative Hamming distance $\Delta(\mathcal{C}_0)$.*

There is a soft error reduction algorithm that for each $n \in \mathbb{N}$, a message $x \in \{0,1\}^n$ and a noise parameter $\varepsilon \in [0, 1-2\Delta(\mathcal{C}_0))$, given $w \in \{0,1\}^{n/r}$ s.t. $\Delta(w, C(x)) \le \frac{1}{4} - \varepsilon$ for $C(x) \in \mathcal{C}(\mathcal{H}, \mathcal{C}_0)$ the encoding of x in the binary ABNNR code, outputs $z \in \{0,1\}^n$ s.t.
$$\Delta(z,x) \le (\sqrt{2}\lambda)^{1/3}$$
and its running time is $O\left(n \cdot 2^{d^2}\right)$.

Remark 6.44. *The theorem extends to codes obtained when replacing the balanced expander graph $H = ([n],[n],E)$ with an unbalanced expander graphs $H = ([k],[b],E)$ with left degree d_L, as long as as the following the expander mixing property holds: For every $A \subseteq [k]$ and $B \subseteq [b]$,*
$$\left| |E(A,B)| - \frac{d_L\,|A|\,|B|}{b} \right| < \lambda d_L \sqrt{|A|\,|B|}$$

6.3 List Decoding via Learning

In this section we present our *list decoding via learning*: In section 6.3.1 we show that every Fourier concentrated and recoverable code is *combinatorially list decodable*; thus proving part 1 of theorem 6.12. In section 6.3.2 we show that every Fourier concentrated and efficiently recoverable code is *efficiently list decodable*; thus proving part 2 of theorem 6.12. In both cases we address the case of unbalanced binary codes. In section 6.3.3 we present the *concentration and agreement lemma* which is the main lemma we use in the proof of the above results. This lemma shows that if w is close (in Hamming distance) to a codeword or a concentrated codes, then w must share a heavy Fourier coefficient with the close codeword.

Next we present Multiplication codes to which we apply the decoding via learning approach: In section 6.3.4 we show that the Multiplication codes $\mathcal{C}^{\mathcal{P}} = \{(\mathbb{Z}_N, P_N, \mathbb{Z}_N)\}_{N\in\mathbb{N}}$ are Fourier concentrated and efficiently recoverable as long as $\mathcal{P} = \{P_N \colon \mathbb{Z}_N \to \mathbb{C}\}_{N\in\mathbb{N}}$ is a family of efficiently computable and well concentrated functions; thus proving Theorem 6.22. Combined with the above decoding via learning approach, we conclude that $\mathcal{C}^{\mathcal{P}}$ is efficiently list decodable whenever $\mathcal{P}$ is a family of efficiently computable and well concentrated functions. In section 6.3.5 we study the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ presented in Example 6.2, showing that the functions half_N are efficiently computable and well concentrated, and concluding that the codes $\mathcal{C}^{\mathsf{half}}$ are $(\frac{1}{2} - \varepsilon, poly(\log|G|, 1/\varepsilon))$-locally list decodable; thus proving Corollary 6.24.

Notations and Terminology. We use the following notation to address the bias in frequency of 0 and 1 symbols in binary codes: For $C \colon G_N \to \{0,1\}$ we denote the fraction of entries whose value is the less frequent symbol out of $1, 0$ by

$$\mathsf{minor}_C = \min_{b\in\{0,1\}} \frac{1}{|G_N|} |\{s \in G_N \mid C(s) = b\}|$$

and denote the fraction of entries with the more frequent symbol by $\mathsf{maj}_C = 1 - \mathsf{minor}_C$. For example, for balanced codes, $\mathsf{minor}_C = \mathsf{maj}_C = \frac{1}{2}$ for any codeword C.

We extend the notion of a *recovery algorithm* to include algorithms receiving a *list* of coefficients instead of a single coefficient: Recall that a recovery algorithm is an algorithm that given $N \in \mathbb{N}, \tau \in \mathbb{R}$ and $0 \neq \beta \in G_N$ returns a list $L_\beta \supseteq \mathsf{InvHeavy}_{N,\tau,\beta}(\mathcal{C})$, and its running time is $poly(\log|G_N|/\tau)$. In the following we slightly abuse terminology naming a "recovery algorithm" also algorithms that receive as input a *list* L' of $\beta \neq 0$'s in G_N (instead of a single β), and outputs the union $L = \bigcup_{0\neq\beta\in L'} L_\beta$ of the output lists L_β. The running time of this recovery algorithm $|L'|\,T$ for $T = poly(\log|G_N|/\tau)$ the running time of the recovery algorithm in Definition 6.11.

6.3.1 Combinatorial List Decoding Bound: Proof of Theorem 6.12 Part 1

We show that every Fourier concentrated and recoverable code is combinatorially list decodable; thus proving part 1 of theorem 6.12. We address the case of unbalanced binary codes; for balanced codes, each occurrence of minor_C in the theorem statement and its proof below is replaced by $\frac{1}{2}$.

Theorem 6.12 part 1 for unbalanced binary codes. Let $\mathcal{C} = \{C_{N,x} \colon G_N \to \mathbb{C}\}_{N\in\mathbb{N},x\in D_N}$ be a concentrated and recoverable code. Then for any codeword $C_{N,x} \in \mathcal{C}$ and any $\varepsilon \in [0, \mathsf{minor}_{C_{N,x}})$,

$$\left|\mathcal{B}_{\mathcal{C}}(C_{N,x}, \mathsf{minor}_{C_{N,x}} - \varepsilon)\right| \leq poly(\log|G_N|/\varepsilon)$$

for $\mathcal{B}_{\mathcal{C}}(C_{N,x}, \mathsf{minor}_{C_{N,x}} - \varepsilon) = \left\{C \in \mathcal{C} \mid \Delta(C, C_{N,x}) \leq \mathsf{minor}_{C_{N,x}} - \varepsilon\right\}$ the set of codewords in $\mathcal{C}$ in the Hamming ball of radius $\mathsf{minor}_{C_{N,x}} - \varepsilon$ around $C_{N,x}$.

Proof. Fix $0 \leq \varepsilon < \mathsf{minor}_{C_{N,x}}$. By Corollary 6.47 for each $C_{N,x}$ and every $C_{N,y} \in \mathcal{B}_{\mathcal{C}}(C_{N,x}, \mathsf{minor}_{C_{N,x}} - \varepsilon)$, $C_{N,x}, C_{N,y}$ share a (non zero) heavy Fourier coefficient:

$$\exists\alpha \neq 0,\ \alpha \in \mathsf{Heavy}_\tau(C_{N,x}) \cap \mathsf{Heavy}_\tau(C_{N,y})$$

for $\tau = poly(\varepsilon/\log|G_N|)$.

For each $\alpha \in G_N$, denote $L_\alpha = \{z \mid \alpha \in \mathsf{Heavy}_\tau(C_{N,z})\}$. Observe that

$$\left|\mathcal{B}_{\mathcal{C}}(C_{N,x}, \mathsf{minor}_{C_{N,x}} - \varepsilon)\right| \leq \sum_{\alpha\in\mathsf{Heavy}_\tau(C_{N,x})} |L_\alpha|$$

To bound the cardinality of the ball, we therefor bound $|L_\alpha|$ and $|\mathsf{Heavy}_\tau|$: Since $\mathcal{C}$ is recoverable, then $|L_\alpha| \leq poly(\log|G_N|/\tau)$. By Parseval Identity and since $\|C_{N,x}\|_2^2 \leq 1$, then $|\mathsf{Heavy}_\tau(C_{N,x})| \leq \|C_{N,x}\|_2^2/\tau \leq 1/\tau$. Combining both bounds these bounds while assigning $\tau = poly(\varepsilon/\log|G_N|)$, we conclude that

$$\left|\mathcal{B}_{\mathcal{C}}(C_{N,x}, \mathsf{minor}_{C_{N,x}} - \varepsilon)\right| \leq poly(\log|G_N|/\varepsilon)$$

□

6.3.2 List Decoding Algorithm: Proof of Theorem 6.12 Part 2

We show that every Fourier concentrated and efficiently recoverable code is efficiently list decodable; thus proving part 2 of theorem 6.12. We address the case of unbalanced binary codes; for balanced codes each occurrence of minor_C in the theorem statement and its proof below is replaced by $\frac{1}{2}$.

Theorem 6.12 part 1 for unbalanced binary codes. Let $\mathcal{C} = \{C_{N,x} \colon G_N \to \mathbb{C}\}_{N \in \mathbb{N}, x \in D_N}$ be a concentrated and efficiently recoverable code over a learnable domain $\mathcal{G} = \{G_N\}_{N \in \mathbb{N}}$. Then $\mathcal{C}$ is $(d - \varepsilon, poly(\log |G_N| / \varepsilon))$-list decodable (in the Hamming distance) for any $\varepsilon \in [0, d)$ where $d = \min_{C \in \mathcal{C}} \mathsf{minor}_C$

Proof. Let $w \colon G_N \to \mathbb{C}$ be a corrupted codeword, we show a list decoding Algorithm that outputs all codewords $C_{N,x}$ s.t. $\Delta(w, C_{N,x}) < \mathsf{minor}_{C_{N,x}} - \varepsilon$ in time $poly(\log |G_N|, 1/\varepsilon)$. By Corollary 6.47 if $C_{N,x}$ is close to w, then there exists $\alpha_0 \neq 0$ which is τ_0-significant both for C_x and for w, where

$$\tau_0 = poly(\varepsilon, 1/\log |G_N|)$$

Denote by Learn the learning algorithm implied by the fact that $\mathcal{G}$ is learnable.[26] Denote by $\mathsf{Recover}$ the recovery algorithm implied by the fact that $\mathcal{C}$ is recoverable.[27]

With τ_0, Learn and $\mathsf{Recover}$ as defined above, here is the list decoding algorithm:

Algorithm 6.45. List decoding via Learning
Input: *Query access to a corrupted codeword* $w \colon G_N \to \mathbb{C}$
Output: *A list of all* $C_{N,x}$ *s.t.* $\Delta(C_{N,x}, w) < \mathsf{minor}_{C_{N,x}} - \varepsilon$
Steps:

1. *Apply the learning algorithm* Learn *on input* N, τ_0 *and query access to* w*; denote its output by* L'.

2. *Apply the recovery algorithm* $\mathsf{Recover}$ *on input* L'*; denote its output by* L.

3. *Output* L

We show that the output of Algorithm 6.45 is indeed all codewords close to the input w. Let C_x be s.t. $\Delta(C_x, w) \leq \mathsf{minor}_{C_x} - \varepsilon$. We show that $x \in L$. By the above, $\exists \alpha_0 \neq 0,\ \alpha_0 \in \mathsf{Heavy}_{\tau_0}(C_{N,x}) \cap \mathsf{Heavy}_{\tau_0}(w)$. Since $\mathcal{G}$ is a learnable domain, $\alpha_0 \in \mathsf{Heavy}_{\tau_0}(w)$ implies that $\alpha_0 \in L'$. Since $\mathcal{C}$ is a recoverable code and $\alpha_0 \in \mathsf{Heavy}_{\tau_0}(C_{N,x}) \cap L'$, then $x \in L$.

Finally, by the definitions of learnable domains and efficiently recoverable codes, clearly Algorithm 6.45 runs in time $poly(\log |G_N|, 1/\varepsilon)$. □

6.3.3 Concentration and Agreement Lemma

The main lemma used in the above proofs is the following "concentration and agreement lemma". This lemma shows that if w is close (in Hamming distance) to a codeword C of a concentrated code, then w must share a heavy Fourier coefficient with C.

[26] That is, given query access to a function $w \colon G_N \to \mathbb{C}$ and a threshold τ, Learn outputs a list $L' \supseteq \mathsf{Heavy}_\tau(w)$ containing finding all τ-significant Fourier coefficients of w, and its running time is $poly(\log |G_N|, 1/\tau)$.

[27] That is, given a list L' of $\beta \neq 0$'s in G_N, $\mathsf{Recover}$ outputs a list $L \supseteq \bigcup_{0 \neq \beta \in L'} \mathsf{InvHeavy}_{N,\tau,\beta}(\mathcal{C})$ for $\mathsf{InvHeavy}_{N,\tau,\beta}(\mathcal{C}) = \left\{ x \in D_N \mid \left|\widehat{C_{N,x}}(\beta)\right|^2 \geq \tau \right\}$, and its running time is $|L'|\, poly(\log |G_N| / \tau)$.

Lemma 6.46 (Concentration and Agreement)**.** *Let $f, g\colon G_N \to \mathbb{C}$ s.t. $\|f\|_2, \|g\|_2 \leq 1$, f is Fourier concentrate, and $\langle f, g\rangle > \varepsilon + \widehat{f}(0)\widehat{g}(0)$ for some $\varepsilon > 0$. Then, there exists an (explicit) threshold τ which is polynomial in $\varepsilon, 1/\log|G_N|$ such that*

$$\exists \alpha \neq 0, \alpha \in \mathsf{Heavy}_\tau(f) \cap \mathsf{Heavy}_\tau(g)$$

Proof. Let Γ be a set of characters on which f is concentrated to within $o(\varepsilon)$ – namely, so that, if we denote by $\bar{\Gamma}$ its complement, it is the case that $\varepsilon' = \|f_{|\bar{\Gamma}}\|_2 \leq o(\varepsilon)$. Note that, by Cauchy-Schwartz

$$\langle f_{|\bar{\Gamma}}, g_{|\bar{\Gamma}}\rangle^2 \leq \|f_{|\bar{\Gamma}}\|_2^2 \cdot \|g_{|\bar{\Gamma}}\|_2^2 \leq \varepsilon'^2 \cdot 1 = \varepsilon'^2$$

and since the Fourier basis is orthonormal

$$\sum_{\alpha\in\Gamma} \widehat{f}(\alpha)\widehat{g}(\alpha) = \langle f_{|\Gamma}, f_{|\Gamma}\rangle \geq \langle f, f\rangle - \left|\langle f_{|\bar{\Gamma}}, f_{|\bar{\Gamma}}\rangle\right| \geq \varepsilon + \left|\widehat{f}(0)\widehat{g}(0)\right| - \varepsilon'$$

This implies that there exists a $\alpha \neq 0$ s.t. $\chi_\alpha \in \Gamma$ and $\widehat{f}(\alpha)\widehat{g}(\alpha) \geq \varepsilon - \frac{\varepsilon'}{|\Gamma|}$. Now, as both $\widehat{f}(\alpha), \widehat{g}(\alpha) \leq 1$, it must be that both $\widehat{f}(\alpha), \widehat{g}(\alpha) \geq \varepsilon - \frac{\varepsilon'}{|\Gamma|} = \tau$. □

Corollary 6.47. *Let $\mathcal{C} = \{C_{N,x}\colon G_N \to \{\pm 1\}\}_{N\in\mathbb{N}, x\in D_N}$ be a Fourier concentrated binary code, and let $C_{N,x}, w\colon G_N \to \{\pm1\}$ s.t. $\Delta(C_{N,x}, w) < \mathsf{minor}_{C_{N,x}} - \varepsilon$. There there exists $\tau = poly(\varepsilon/\log|G_N|)$ s.t. $\exists\alpha \neq 0$, $\alpha \in \mathsf{Heavy}_\tau(C_{N,x}) \cap \mathsf{Heavy}_\tau(w)$.*

Proof. We first show that for any $C_{N,x}, w$, if $\Delta(C_{N,x}, w) < \mathsf{minor}_{C_{N,x}} - \varepsilon$ then $\langle C_{N,x}, w\rangle > 2\varepsilon + \left|\widehat{C_{N,x}}(0)\widehat{w}(0)\right|$. This is because for binary codes, $\langle C_{N,x}, w\rangle = 1 - 2\Delta(C_{N,x}, w)$, and because $1-2(\mathsf{minor}_{C_{N,x}} - \varepsilon) = \mathsf{maj}_{C_{N,x}} - \mathsf{minor}_{C_{N,x}} = \left|\widehat{C_{N,x}}(0)\right| \geq \left|\widehat{C_{N,x}}(0)\widehat{C_{N,y}}(0)\right|$ where the last inequality is true since $|\widehat{w}(0)| \leq 1$ for any Boolean w.

Thus, by Concentration and Agreement Lemma 6.46, since $\mathcal{C}$ is concentrated, then for each $C_{N,x}$ and every $w \in \mathcal{B}(C_{N,x}, \mathsf{minor}_{C_{N,x}} - \varepsilon)$,

$$\exists\alpha \neq 0,\ \alpha \in \mathsf{Heavy}_\tau(C_{N,x}) \cap \mathsf{Heavy}_\tau(C_{N,y})$$

for $\tau = poly(\varepsilon/\log|G_N|)$. □

6.3.4 List Decoding Codes $\mathcal{C}^{\mathcal{P}}$ for Well Concentrated $\mathcal{P}$: Proof of Theorem 6.22

We show that codes $\mathcal{C}^{\mathcal{P}} = \{(\mathbb{Z}_N, P_N, \mathbb{Z}_N)\}_{N\in\mathbb{N}}$ with $\mathcal{P} = \{P_N\colon \mathbb{Z}_N \to \mathbb{C}\}_{N\in\mathbb{N}}$ a family of efficiently computable and well concentrated functions[28] are Fourier concentrated and efficiently recoverable; thus proving Theorem 6.22. By Theorem 6.12 Part 2 combined with the learnability of $\mathbb{Z}_N$ proved in Theorem 3.16, this implies that the

[28]Recall that a function is *well concentrated* if it is concentrated on a small set of characters with low gcd (greatest common divisor) with N.

codes $\mathcal{C}^{\mathcal{P}}$ are locally list decodable when $\mathcal{P}$ is a family of efficiently computable and well concentrated functions.

Theorem 6.22 Let $\mathcal{P}$ be a family of well concentrated functions, then the Multiplication codes $\mathcal{C}^{\mathcal{P}} = \{(\mathbb{Z}_N, P_N, \mathbb{Z}_N)\}_{N\in\mathbb{N}}$ are Fourier concentrated and recoverable. If the functions in $\mathcal{P}$ are efficiently computable (in addition to being Fourier concentrated and recoverable), then $\mathcal{C}^{\mathcal{P}}$ is *efficiently* recoverable.

Proof. In Lemma 6.48 we show that $\mathcal{C}^{\mathcal{P}}$ is Fourier concentrated. In Lemma 6.49 we show that $\mathcal{C}^{\mathcal{P}}$ is efficiently recoverable.

Lemma 6.48 ($\mathcal{C}^{\mathcal{P}}$ is concentrated)**.** *If $\mathcal{P}$ is Fourier concentrated, then the multiplication code $\mathcal{C}^{\mathcal{P}}$ is Fourier concentrated.*

Proof. For all $N \in \mathbb{N}$, and all codewords $C_{N,x} \in \mathcal{C}^{\mathcal{P}}$, recall that $C_{N,x}(j) = P_N(jx \mod N)$. Since $\mathcal{P}$ is Fourier concentrated, the for all $\varepsilon > 0$, $\exists \Gamma \subseteq \mathbb{Z}_N$ of size $poly(\log N/\varepsilon)$ s.t. $\|P_N - P_{N|\Gamma}\|_2^2 \leq \varepsilon$. By Theorem 2.2 in section 2, this implies that $\|C_{N,x} - C_{N,x|\Gamma'}\|_2^2 \leq \varepsilon$ for

$$\Gamma' = \{\alpha/x \mid \alpha \in \Gamma\}$$

Thus $\mathcal{C}^{\mathcal{P}}$ is concentrated. □

Lemma 6.49 ($\mathcal{C}^{\mathcal{P}}$ is efficiently recoverable)**.** *If $\mathcal{P}$ is a family of efficiently computable and well concentrated functions, then the multiplication code $\mathcal{C}^{\mathcal{P}}$ is efficiently recoverable.*

Proof. In Lemma 6.49 we showed that for every $N \in \mathbb{N}$, $\tau \in \mathbb{R}^+$ and $0 \neq \beta \in \mathbb{Z}_N$, the set

$$L = \bigcup_{\alpha \in \mathsf{Heavy}_\tau(P_N)} \{x \mid \beta \equiv \alpha x^{-1} \mod N\}$$

satisfies that $L \supseteq \mathsf{InvHeavy}_{N,\tau,\alpha}(\mathcal{C}^{\mathcal{P}})$ and $|L| \leq poly(\log N/\tau)$. Thus an algorithm that on input $N, \tau, \beta \neq 0$ outputs a short list containing L in running time $poly(\log N/\tau)$ is an efficient recovery algorithm. We show such an algorithm below.

Algorithm 6.50. Recovery Algorithm for $\mathcal{C}^{\mathcal{P}}$.
Input: $N \in \mathbb{N}$, $\tau \in \mathbb{R}^+$ *and* $0 \neq \beta \subseteq \mathbb{Z}_N$
Output: $L \supseteq \{x \mid \beta \in \mathsf{Heavy}_\tau(C_{N,x})\}$

```
1. Run SFT Algorithm 3.4 on input $N, \tau$ and query access to $P_N$; denote its output by $H$.
2. For each $\alpha \in H$
    (a) Run Euclid Algorithm to find $d = gcd(\alpha, N)$
    (b) Run Extended Euclid Algorithm to find $x \in \mathbb{Z}_N$ s.t.
```

$$\beta \equiv \alpha x^{-1} \mod \frac{N}{d}$$

(c) *Let* $L_\alpha = \left\{x + i \cdot \frac{N}{d} \mod N)\right\}_{i=0,\dots,d-1}$

3. *Output* $L' = \bigcup_{\alpha \in H} L_\alpha$

We show that the output L' of the Recovery Algorithm 6.50 satisfies that $L' \supseteq L$ and $|L'| \leq poly(\log N/\tau)$. We first argue that $L' \supseteq L$. Observe that

$$L' = \bigcup_{\alpha \in H} \left\{x \mid \beta \equiv \alpha x^{-1} \mod N\right\} \supseteq L$$

where the last containment is true because by Theorem 3.16, the output H of the SFT algorithm satisfies that $H \supseteq \mathsf{Heavy}_\tau(P_N)$ and by definition of L. We next argue that L' is of size at most $poly(\log N, 1/\tau)$. Since $\mathcal{P}$ is well concentrated, then the value d in step 2b is at most $poly(\log N/\tau)$, implying that $\forall \alpha \in H$, the set L_α satisfies that $|L_\alpha| = d \leq poly(\log N/\tau)$. Since by Theorem 3.16, $|H| \leq O(1/\tau)$, we conclude that

$$|L'| \leq poly(\log N/\tau)$$

We show that the running time of the above algorithm is $poly(\log N/\tau)$. Since P_N is efficiently computable then we can provide query access to P_N in time $poly \log N$, and by Theorem 3.16 the running time of step 1 of the algorithm is thus $poly(\log N/\tau)$. Since $\mathcal{P}$ is well concentrated, then $d \leq poly(\log N/\tau)$, implying, by efficiency of the Extended Euclid Algorithm, that the running time of step 2 is $poly(\log N/\tau)$. Combined together, we conclude that the algorithm runs in time $poly(\log N/\tau)$. □

Remark 6.51. *The requirement that $\mathcal{P}$ is a family of* efficiently computable *functions can sometimes be relaxed. This assumptions is used only for computing $H \supseteq \mathsf{Heavy}_\tau(P_N)$. Observe that $\mathsf{Heavy}_\tau(P_N)$ does not depend on the corrupted codeword received by the list decoding algorithm, but rather it is a global information about the code $\mathcal{C}^{\mathcal{P}}$. Therefore, it may be reasonable to assume that $\mathsf{Heavy}_\tau(P_N)$ is given as prior knowledge, in which case we do not need to be able to compute it efficiently. In this case, there's no need to require that $\mathcal{P}$ is efficiently computable.*

6.3.5 List Decoding $\mathcal{C}^{\mathsf{half}}$: Proof of Corollary 6.24

We show that the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ presented in Example 6.2 are is $(\frac{1}{2} - \varepsilon, poly(\log|G|, 1/\varepsilon))$-locally list decodable; thus proving Corollary 6.24.

Corollary 6.24 For any $N \in \mathbb{N}$, the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ presented in Example 6.2 are is $(\frac{1}{2} - \varepsilon, poly(\log|G|, 1/\varepsilon))$-locally list decodable.

Proof. The corollary follows from Theorem 6.22 above saying that $\mathcal{C}^{\mathcal{P}}$ is list decodable if $\mathcal{P}$ is well concentrated when combined with the lemma below showing that the family of functions $\mathsf{Half} = \{\mathsf{half}_N \colon \mathbb{Z}_N \to \{0,1\}\}_{N \in \mathbb{N}}$ is efficiently computable and well concentrated. □

Lemma 6.23 The family of functions $\mathsf{Half} = \{\mathsf{half}_N \colon \mathbb{Z}_N \to \{0,1\}\}_{N\in\mathbb{N}}$ defined by $\mathsf{half}_N(x) = 1$ iff $\min\{x, N-x\} \leq N/4$ is efficiently computable and well concentrated.

Proof. The fact that for Half is efficiently computable is immediate from its definition. We show that Half is well concentrated.

Fix $N \in \mathbb{N}$ and $\varepsilon > 0$, and drop indexes N. To show that Half is well concentrated, we show there is a set of characters $\Gamma \subseteq \mathbb{Z}_N$ of size $|\Gamma| = poly(\log N, 1/\varepsilon)$, of low gcd $\forall \alpha \in \Gamma$, $gcd(\alpha, N) \leq poly(\log N, 1/\varepsilon)$, s.t. $\| \sum_{\alpha\in\Gamma} \widehat{\mathsf{half}}(\alpha)\chi_\alpha - \mathsf{half}\|_2^2 \leq \varepsilon$. Set

$$\Gamma = \{\alpha \mid \mathsf{abs}(\alpha) \leq k\} \text{ for } k = O(1/\varepsilon)$$

Clearly, $|\Gamma| = O(1/\varepsilon)$ and $\forall \alpha \in \Gamma$, $gcd(\alpha, N) \leq k = O(1/\varepsilon)$. To show that $\| \sum_{\alpha\in\Gamma} \widehat{\mathsf{half}}(\alpha)\chi_\alpha - \mathsf{half}\|_2^2 \leq \varepsilon$ it suffices to show that $\sum_{\alpha\notin\Gamma} \left|\widehat{\mathsf{half}}(\alpha)\right|^2 \leq \varepsilon$. For each $0 \neq \alpha \in \mathbb{Z}_N$, $\widehat{\mathsf{half}_N}(\alpha) = S_{N/2}(\alpha)$ (for $S_{N/2}(\alpha)$ as in Definition 3.32). By Proposition 3.33 Item 3, $\left|S_{N/2}(\alpha)\right|^2 \leq \frac{2}{3}\left(\frac{2}{\mathsf{abs}(\alpha)}\right)^2$. Therefore,

$$\sum_{\alpha\notin\Gamma} \left|\widehat{\mathsf{half}}(\alpha)\right|^2 = O\left(\sum_{\alpha\geq k}\frac{1}{\alpha^2}\right) = O\left(\frac{1}{k}\right)$$

which is equal to ε by choice of k. □

6.4 Distance Bounding using Fourier Spectrum: Proof of Theorem 6.27

We analyze the combinatorial properties of codes $\mathcal{C}^{P_N} = (\mathbb{Z}_N, P_N, \mathbb{Z}_N)$. We bound the distance of $\mathcal{C}^{P_N}$ in terms of the Fourier spectrum of P_N and analyze its rate, showing it achieves parameters as stated in Theorem 6.27. We then focus on the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ presented in Example 6.2, showing they have constant distance. This proves Corollary 6.28.

To enable giving our distance bound, we slightly restrict the message space of the considered codes to be the set

$$M_N = \mathbb{Z}_N^* \cap \{1, \ldots, \lfloor (N/2) \rfloor\}$$

This restriction incurs only a small reduction in size of the message space.

Theorem 6.27. For any $N \in \mathbb{N}$ and $P_N \colon \mathbb{Z}_N \to \{\pm 1\}$, the MPC code $(\mathbb{Z}_N, P_N, \mathbb{Z}_N)$ is a code of rate $(\log N - O(1))/N$ and distance at least

$$\left(\left|\widehat{P_N}(\alpha_1)\right| - \left|\widehat{P_N}(\alpha_2)\right|\right)^2$$

for $\alpha_1, \alpha_2 \in \mathbb{Z}_N$ defined by $\alpha_1 = \arg\max_{\alpha \in \mathbb{Z}_N} \left|\widehat{P_N}(\alpha)\right|$ and $\alpha_2 = \arg\max_{\alpha \in \mathbb{Z}_N, \alpha \neq \pm\alpha_1} \left|\widehat{P_N}(\alpha)\right|$ where $\widehat{P_N}(\alpha) = \mathbb{E}_{x \in \mathbb{Z}_N}\left[P_N(x) e^{i\frac{2\pi}{N}\alpha x}\right]$.

Proof. We first bound the rate. The rate is $(\log N - O(1))/N$ because the information length is $\log|M_N| = \log N - \log\Theta(1)$, whereas the encoding length is N.

We next bound the distance. Denote $\tau_1 = \left|\widehat{P_N}(\alpha_1)\right|^2$ and $\tau_2 = \left|\widehat{P_N}(\alpha_2)\right|^2$. We show that the distance $\Delta(C_x, C_y)$ is at least $(\sqrt{\tau_1} - \sqrt{\tau_2})^2$ for any two codewords $C_x, C_y \in \mathcal{C}^{P_N}$. Recall that

$$\Delta(C_x, C_y) = \frac{1}{2} - \frac{1}{2}\langle C_x, C_y\rangle$$

therefore to lower bound the distance between C_x and C_y we may as well upper bound their inner product. By Parseval Identity (see Theorem 2.2),

$$\langle C_x, C_y\rangle = \sum_{\alpha \in \mathbb{Z}_N} \widehat{C_x}(\alpha)\widehat{C_y}(\alpha)$$

Observe that $\widehat{C_x(\alpha)} = \widehat{P_N(\alpha/x)}$ and $\widehat{C_y(\alpha)} = \widehat{P_N(\alpha/y)}$. (This is because $\forall z \in \mathbb{Z}_N^*$, $C_z(i) = P_N(i/z \mod N)$ and by Theorem 2.2.) Therefore,

$$\langle C_x, C_y\rangle = \sum_{\alpha \in \mathbb{Z}_N} \widehat{P_N}(\alpha)\widehat{P_N}(\alpha\frac{x}{y})$$

$\langle C_x, C_y\rangle$ is therefore maximal when largest Fourier coefficients of P_N are multiplied with each other. Since $y \in M_N$, then $x \neq \pm y$, implying that $\alpha_1 \notin \left\{\pm\alpha_1 \cdot \frac{x}{y} \mod N\right\}$; therefore, at the worst case: $\alpha_1 \in \left\{\pm\alpha_2 \cdot \frac{x}{y} \mod N\right\}$ and $\alpha_2 \in \left\{\pm\alpha_1 \cdot \frac{x}{y} \mod N\right\}$. In this case,

$$\langle C_x, C_y\rangle = 4\sqrt{\tau_1\tau_2} + \sum_{\alpha \neq \pm\alpha_1, \pm\alpha_2} \widehat{P_N}(\alpha)\widehat{P_N}(\alpha\frac{x}{y})$$

Observing that $\sum_{\alpha \neq \pm\alpha_1, \pm\alpha_2} \widehat{P_N}(\alpha)\widehat{P_N}(\alpha\frac{x}{y}) \leq \sum_{\alpha \neq \pm\alpha_1, \pm\alpha_2} \left|\widehat{P_N}(\alpha)\right|^2 \leq 1 - 2\tau_1 - 2\tau_2$ and using the identify $(a-b)^2 = a^2 - 2ab + b^2$ we get that

$$\langle C_x, C_y\rangle \leq 1 - 2(\sqrt{\tau_1} - \sqrt{\tau_2})^2$$

Therefore

$$\Delta(C_x, C_y) \geq (\sqrt{\tau_1} - \sqrt{\tau_2})^2$$

□

We show the codes $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ have constant distance. To show this we analyze the Fourier spectrum of the functions half_N bounding from below their heaviest Fourier coefficient by $\frac{2}{\pi}$, and bounding from above their second heaviest Fourier coefficient by $\frac{2}{3\pi}\left(1 + O\left(\frac{1}{N}\right)\right)$.

Corollary 6.28. For any $N \in \mathbb{N}$, the Multiplication code $(\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ with message space M_N is a binary code of rate $(\log N - O(1))/N$ and distance at least $\left(\frac{4}{3\pi}\right)^2\left(1 - O\left(\frac{1}{N}\right)\right) \approx 0.18$.

Proof. The fact that $\mathcal{C}^{\mathsf{half}_N}$ is binary is immediate from Half being a family of Boolean function. The rate bound follows from N being the encoding length and $\log|M_N| = \log O(N)$ being the information rate. In the Lemma below we show that $(\sqrt{\tau_1} - \sqrt{\tau_2})^2 > \left(\frac{4}{3\pi}\right)^2\left(1 - O\left(\frac{1}{N}\right)\right)$ for τ_1, τ_2 the largest and second largest values of $\left|\widehat{\mathsf{half}_N}(\alpha)\right|^2$ over all $\alpha \in \mathbb{Z}_N, N \in \mathbb{N}$. By Theorem 6.27 this implies that $\mathcal{C}^{\mathsf{half}_N}$ has distance of at least $\left(\frac{4}{3\pi}\right)^2\left(1 - O\left(\frac{1}{N}\right)\right)$. $\square$

Lemma 6.52. *Denote τ_1, τ_2 the largest and second largest values of $\left|\widehat{\mathsf{half}_N}(\mathsf{abs}(\alpha))\right|^2$ over all $\alpha \in \mathbb{Z}_N, N \in \mathbb{N}$, then $\tau_1 \geq \left(\frac{2}{\pi}\right)^2$, $\tau_2 \leq \left(\frac{2/3}{\pi}\right)^2\left(1 + O(\frac{1}{N})\right)$ and $(\sqrt{\tau_1} - \sqrt{\tau_2})^2 \geq \left(\frac{4}{3\pi}\right)^2 - O(\frac{1}{N})$*

Proof. Fix N. We analyze to Fourier coefficients of half_N, lower bounding the largest coefficient τ_1 by $(2/\pi)^2$ and upper bounded the second largest coefficient τ_2 by $\left(\frac{2/3}{\pi}\right)^2\left(1 + O(\frac{1}{N})\right)$. Combing these bounds we conclude that $(\sqrt{\tau_1} - \sqrt{\tau_2})^2 > \left(\frac{4}{3\pi}\right)^2 - O(\frac{1}{N})$.

We first argue that the largest Fourier coefficient is obtained on $\mathsf{abs}(\alpha) = 1$ and the second largest on $\mathsf{abs}(\alpha) = 3$. For $\alpha = 0$, $\widehat{\mathsf{half}_N}(0) \leq \frac{1}{N}$ so its negligibly small. We concentrate of $\alpha \neq 0$. For each $0 \neq \alpha \in \mathbb{Z}_N$, the α Fourier coefficients of half_N is equal to $S_{N/2}(\alpha)$ (for $S_{N/2}(\alpha)$ as presented in Definition 3.32), which by proposition 3.33 Item 1, implies that

$$|\mathsf{half}_N(\alpha)|^2 = \left(\frac{2}{N}\right)^2 \frac{1 - \cos(\frac{2\pi}{N}\alpha\frac{N}{2})}{1 - \cos(\frac{2\pi}{N}\alpha)}$$

Exploring this function reveals that it has local maxima on odd α with decreasing value as $\mathsf{abs}(\alpha)$ increase. Namely, the largest Fourier coefficient is obtained on α s.t. $\mathsf{abs}(\alpha) = 1$, and the second largest is obtained on α s.t. $\mathsf{abs}(\alpha) = 3$.

We next lower bound the largest Fourier coefficient of half_N by $(2/\pi)^2$, showing that $|\mathsf{half}_N(1)|^2 > (2/\pi)^2$. To simplify the computations, we restrict attention to even values of N. For odd N, results differ by negligible factors of $O(\frac{1}{N})$. By the above, when assigning $\alpha = 1$, $\left|\widehat{\mathsf{half}_N}(1)\right|^2 = \left(\frac{2}{N}\right)^2 \frac{1 - \cos\pi}{1 - \cos(\frac{2\pi}{N})}$. The nominator is equal to 2. The denominator is upper bounded by $\frac{1}{2}\left(\frac{2\pi}{N}\right)^2$ (according to Taylor approximation). Combining these two bound we conclude that

$$\tau_1 \geq \left(\frac{2}{\pi}\right)^2$$

We next upper bound τ_2. By the above, $\left|\widehat{\mathsf{half}_N}(3)\right|^2 = \left(\frac{2}{N}\right)^2 \frac{1 - \cos(3\pi)}{1 - \cos(\frac{2\pi}{N}3)}$ The nominator is equal to 2. The denominator is lower bounded by $\frac{1}{2}(\frac{2\pi}{N}3)^2 - \frac{1}{4!}(\frac{2\pi}{N}3)^4$ (according to Taylor approximation). Reorganizing this expression we get that $\left|\widehat{\mathsf{half}_N}(3)\right|^2 \leq \left(\frac{2/3}{\pi}\right)^2 \frac{1}{1-3\left(\frac{\pi}{N}\right)^2}$. Namely,

$$\tau_2 \leq \left(\frac{2/3}{\pi}\right)^2 \left(1 + O(\frac{1}{N})\right)$$

Combining the two bounds, we get that the distance of $\mathcal{C}^{\mathsf{Half}}$ is at least

$$(\sqrt{\tau_1} - \sqrt{\tau_2})^2 \geq \left(\frac{4}{3\pi}\right)^2 - O(\frac{1}{N})$$

□

6.5 Self Correcting via Testing: Proof of Theorem 6.25

We show that the Multiplication codes $\mathcal{C}^{\mathsf{half}}$ from Example 6.2 are locally self correctable. This proves Theorem 6.25.

Theorem 6.25. Let $\mathcal{C}^{\mathsf{half}} = (\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ be the Multiplication code as in Example 6.2, denote $\tau_{max} = \max_{\alpha\in\mathbb{Z}_N} \left|\widehat{\mathsf{half}_N}(\alpha)\right|^2$. Then $\tau_{max} = \frac{1}{4} + \Omega(1)$ and for every $\varepsilon \in [0, \tau_{max} - \frac{1}{4})$ and $\rho \in (0,1)$, $\mathcal{C}$ is $(\varepsilon, q, T, 1-\rho)$-locally self correctable with number of queries bounded by $q \leq poly(1/(\tau_{max} - \frac{1}{4} - \varepsilon), 1/\rho)$ and running time bounded by $T \leq poly(\log|G|, 1/(\tau_{max} - \frac{1}{4} - \varepsilon), 1/\rho)$.

Proof. To prove Theorem 6.25, we first present our local self correcting algorithm, and then analyze its correctness and complexity.

Algorithm 6.53. Local Self Correcting Algorithm.
Input: *$N \in \mathbb{N}$, entry location $y \in \mathbb{Z}_N$, noise parameter $\varepsilon \in [0, \tau_{max} - \frac{1}{4})$, success parameter $\rho \in (0,1)$, and query access to $w\colon \mathbb{Z}_N \to \{\pm 1\}$ s.t. $\|w - C_{N,x}\|_2^2 \leq \varepsilon$ for a codeword $C_{N,x} \in \mathcal{C}^{\mathsf{half}}$.*
Output: $C_{N,x}(y)$
Steps:

```
1. Denote $\zeta = \tau_{max} - \frac{1}{4} - \frac{\varepsilon}{2}$,  and $\ell = \Theta(\zeta^{1.5}\rho N)$
2. For $c = i\ell$ for $i = 0, \ldots, \lfloor(\frac{1}{\ell}(1-\rho)\frac{N}{4})\rfloor$
    (a) Denote $\delta = O(\frac{\zeta^{1.5}\rho}{1-\rho})$, $m_A = \Theta\left(\frac{1}{\zeta^2}\ln\frac{1}{\delta}\right)$, $m_B = \Theta\left(\frac{1}{\zeta^2}(\ln m_A)(\ln\frac{1}{\delta})\right)$ and
        $t = \Theta(\frac{N\sqrt{\zeta}}{\ell})$
```

(b) *Let $A \subseteq \mathbb{Z}_N$ be a set of m_A elements chosen uniformly at random. Let $B \subseteq \{0,\dots,t-1\}$ be a set of $\min\{t, m_B\}$ elements chosen uniformly at random.*

(c) *Run the Parametrized Distinguishing Algorithm 6.55 on input N, c, A, B, $\frac{1}{2}+\zeta$ and query access to the function f as given by the Simulated Query Access Algorithm 6.54; denote its output by $decision_c$*

3. *Output 1 iff $\exists c \in 0,\dots,\lfloor(\frac{1}{\ell}(1-\rho)\frac{N}{4})\rfloor$ for which $decision_c = 1$; output -1 otherwise.*

Algorithm 6.54. Simulated Query Access Algorithm.
Input: *$j \in \mathbb{Z}_N$ the entry location $y \in \mathbb{Z}_N$ and query access to $w\colon \mathbb{Z}_N \to \{\pm 1\}$*
Output: *$f(j)$ for f s.t. $\Delta(C_{N,xy}, f) = \Delta(C_{N,x}, C_{N,x})$*
Steps: *Output $w(yj)$ for y the input to the Local Self Correcting Algorithm and yj multiplication modulo N.*

Algorithm 6.55. Distinguishing Algorithm.
Input: *$N \in \mathbb{N}$, $c \in \mathbb{Z}_N$, $A, B \subseteq \mathbb{Z}_N$, $\tau \in [0,1]$ and query access to a function $f\colon \mathbb{Z}_N \to \mathbb{C}$*
Output: *1 or 0*
Steps:

1. *Compute*

$$\begin{aligned}
\mathsf{est}_+ &\leftarrow \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B}\chi_{-c}(y) f(x-y)\right)^2 \\
\mathsf{est}_- &\leftarrow \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B}\chi_{c}(y) f(x-y)\right)^2 \\
\mathsf{est} &\leftarrow \mathsf{est}_+ + \mathsf{est}_-
\end{aligned}$$

2. *If $\mathsf{est} \geq \tau$, output 1, else output 0*

Analysis of Local Self Correcting Algorithm 6.53. Let the input to the Local Self Correcting Algorithm 6.53 be $N \in \mathbb{N}$, $y \in \mathbb{Z}_N$, $\varepsilon \in [0, \tau_{max} - \frac{1}{4})$, $\rho \in (0,1)$ and query access to $w\colon \mathbb{Z}_N \to \{\pm 1\}$ s.t. $\|w - C_x\|_2^2 \leq \varepsilon$ for $C_x\colon \mathbb{Z}_N \to \{\pm 1\}$ a codeword of $\mathcal{C}^{\mathsf{Half}}$. We show that the Local Self Correcting Algorithm 6.53 ourputs the value $C_x(y)$ with probability at least $2/3$ for at least $1-\rho$ fraction of the possible inputs y, and that its query complexity and running time is $q, T = \widetilde{\Theta}\left(\frac{1-\rho}{\zeta^{5.5}\rho}\ln^2\frac{1}{\rho}\right)$. This shows that for any $\rho \in (0,1)$ and $\varepsilon \in [0, \tau_{max} - \frac{1}{4})$, $\mathcal{C}^{\mathsf{Half}}$ is $(q, T, 1-\rho)$-locally self correctable algorithm from noise ε with $q, T = \widetilde{\Theta}\left(\frac{1-\rho}{\zeta^{5.5}\rho}\ln^2\frac{1}{\rho}\right)$.

Denote

$$\alpha_{max} = \mathsf{abs}(\arg\max_{\alpha\in\mathbb{Z}_N}\left|\widehat{C_{N,xy}}(\alpha)\right|^2)$$

Consider first the case that $C_x(y) = 1$. In this case, by Lemma 6.56, $\alpha_{max} \in \left\{-\frac{N}{4}, \dots, \frac{N}{4}\right\}$ and for $f\colon \mathbb{Z}_N \to \{\pm 1\}$ the function defined by the Simulated Query Access Algorithm 6.54 it holds that $\left|\widehat{f}(\alpha_{max})\right|^2 + \left|\widehat{f}(-\alpha_{max})\right|^2 \geq \frac{1}{2} + \zeta$. We consider two sub-cases: either $\alpha_{max} \in [0, (1-\rho)\frac{N}{4}]$ or $\alpha_{max} \in ((1-\rho)\frac{N}{4}, \frac{N}{4}]$.

In the case when $\alpha_{max} \in [0, (1-\rho)\frac{N}{4}]$, the algorithm outputs the correct value $C_x(y)$, with probability at least 2/3 (where the probability is taken over the random coins of the algorithm). This is because by Lemma 6.57 $decision_c = 1$ with probability at least $1-\delta$ for the $c \in [0, \lfloor((1-\rho)\frac{N}{4})\rfloor]$ s.t. $\mathsf{abs}(\alpha_{max} - c) < \ell/2$, implying that the algorithm outputs value 1 with probability at least $1-\delta$.

In the case when $\alpha_{max} \in ((1-\rho)\frac{N}{4}, \frac{N}{4}]$, we have no guarantee on the output of the algorithm. This case however happens for at most $1-\rho$ fraction of the entries y. This is because $\alpha_{max} = \mathsf{abs}(xy)$ (see Lemma 6.56.2), and $\{xy \mid y \in \mathbb{Z}_N\}$ is a permutation of $\mathbb{Z}_N$ for $x \in \mathbb{Z}_N^*$. Thus, the fraction of y's such that $\mathsf{abs}(xy) \in ((1-\rho)\frac{N}{4}, \frac{N}{4}]$ is at most ρ.

Consider next the case that $C_x(y) = -1$. In this case, by Lemma 6.56, $\sum_{\alpha \in \left\{-\frac{N}{4}, \dots, \frac{N}{4}\right\}} \left|\widehat{f}(\alpha)\right|^2 \leq \frac{1}{2} - \zeta$. By Lemma 6.57, this implies that for every c, $decision_c = 0$, with probability at least $1-\delta$. By union bound this holds for all $O(\frac{1-\rho}{\zeta^{1.5}\rho})$ values of c with probability at least $(1-\delta)^{O(\frac{1-\rho}{\zeta^{1.5}\rho})}$ which is greater than 2/3 by the choice of $\delta \leq O(\frac{\zeta^{1.5}\rho}{1-\rho})$. Thus, the algorithm outputs value -1 with probability at least 2/3.

We conclude by analyzing the number q of queries the algorithm makes and its running time T. The number of repetitions of loop over intervals centers c is at most $\frac{1}{\ell} \cdot (1-\rho)\frac{N}{4} = O(\frac{1-\rho}{\zeta^{1.5}\rho})$. Each run of the Distinguishing Algorithm makes $m_A \cdot m_B = \widetilde{\Theta}((\frac{1}{\zeta^2}\ln\frac{1}{\delta})^2)$ queries and takes time $O(m_A m_B)$. The total number of queries and the total running time are therefore

$$q, T \;=\; \widetilde{\Theta}\left(\frac{1-\rho}{\zeta^{5.5}\rho}\ln^2\frac{1}{\rho}\right)$$

(where we replaced $\ln\frac{1}{\delta}$ with $\ln\frac{1}{\rho}$ because all arguments other than $1/\rho$ are subsumed by the $\widetilde{\Theta}()$ notation for our choice of $\delta = O(\frac{\zeta\rho}{1-\rho})$). □

Lemma 6.56. *For $f\colon \mathbb{Z}_N \to \{\pm 1\}$ the function defined by the Simulated Query Access Algorithm 6.54 the following holds:*

- *If $C_{N,x}(y) = 1$, then $\alpha_{max} \in \left\{-\frac{N}{4}, \dots, \frac{N}{4}\right\}$ and $\left|\widehat{f}(\alpha_{max})\right|^2 + \left|\widehat{f}(-\alpha_{max})\right|^2 \geq \frac{1}{2} + \zeta$.*

- *If $C_{N,x}(y) = -1$, then $\sum_{\alpha \in \left\{-\frac{N}{4}, \dots, \frac{N}{4}\right\}} \left|\widehat{f}(\alpha)\right|^2 \leq \frac{1}{2} - \zeta$.*

Proof. Consider first the case $C_{N,x}(y) = 1$. In this case, by Lemma 6.56.3, $\alpha_{max} \in \left\{-\frac{N}{4}, \dots, \frac{N}{4}\right\}$ and $\left|\widehat{C_{N,xy}}(\alpha_{max})\right|^2 + \left|\widehat{C_{N,xy}}(-\alpha_{max})\right|^2 \geq 2\tau_{max}$. By Lemma 6.56.1, this implies that $\left|\widehat{f}(\alpha)\right|^2 + \left|\widehat{f}(-\alpha)\right|^2 \geq 2\tau_{max} - \varepsilon$. From the definition of ζ it follows that

$$\left|\widehat{f}(\alpha)\right|^2 + \left|\widehat{f}(-\alpha)\right|^2 \geq \frac{1}{2} + \zeta$$

Consider next the case $C_{N,x}(y) = -1$. In this case, by Lemma 6.56.3, $\sum_{\alpha\in\left\{-\frac{N}{4},\dots,\frac{N}{4}\right\}} \left|\widehat{C_{N,xy}}(\alpha)\right|^2 < 1 - 2\tau_{max}$. Lemma 6.56.1, this implies that $\sum_{\alpha\in\left\{-\frac{N}{4},\dots,\frac{N}{4}\right\}} \left|\widehat{f}(\alpha)\right|^2 < 1 - 2\tau_{max} + \varepsilon$. From the definition of ζ it follows that

$$\sum_{\alpha\in\left\{-\frac{N}{4},\dots,\frac{N}{4}\right\}} \left|\widehat{f}(\alpha)\right|^2 < \frac{1}{2} - O(1)$$

Lemma 6.56.1 (Analysis of Simulated Query Access Algorithm 6.54)**.** *Let $f\colon \mathbb{Z}_N \to \{\pm 1\}$ be the function defined by the output of Simulated Query Access Algorithm 6.54, then for all $x \in \mathbb{Z}_N^*$,*

$$\|f - C_{N,xy}\|_2^2 \leq \varepsilon$$

Proof. Let η, η' be s.t. $w = C_{N,x} + \eta$ and $f = C_{N,xy} + \eta'$. By definition, $f(j) = C_{N,x}(yj) + \eta(yj)$. Observe that $C_{N,x}(yj) = \mathsf{half}_N(x \cdot yj) = C_{N,xy}(j)$, and $\|\eta'\|_2^2 = \|\eta\|_2^2$ (where we rely on the fact that the map $j \mapsto yj$ is a permutation of $\mathbb{Z}_N^*$). Finally, since $\|\eta\|_2^2 \leq \varepsilon$ we conclude that $\|f - C_{N,xy}\|_2^2 \leq \varepsilon$. □

Lemma 6.56.2. *$\alpha_{max} = \mathsf{abs}(xy)$ and $C_{N,x}(y) = 1$ iff $\alpha_{max} \in \{\alpha \mid \mathsf{abs}(\alpha) \leq N/4\}$.*

Proof. We first argue that $\alpha_{max} = \mathsf{abs}(xy)$. Recall that by Theorem 2.2,

$$\widehat{C_{N,xy}}(\alpha) = \widehat{\mathsf{half}_N}(\alpha(xy)^{-1})$$

In particular for $\alpha \in \pm xy$,

$$\widehat{C_{N,xy}}(xy) \in \widehat{\mathsf{half}_N}(\pm 1)$$

Since the heaviest Fourier coefficient of half_N is on ± 1, then the heaviest Fourier coefficient of $C_{N,xy}$ is on $\pm xy$. Namely,

$$\alpha_{max} = \mathsf{abs}(xy)$$

We next argue that $C_{N,x}(y) = 1$ iff $xy \in \{\alpha \mid \mathsf{abs}(\alpha) \leq N/4\}$. By definition of $\mathcal{C}$, $C_{N,x}(y) = \mathsf{half}_N(xy)$. By definition of Half, $\mathsf{half}_N(xy) = 1$ iff $\mathsf{abs}(xy) \leq N/4$. Thus,

$$C_{N,x}(y) = 1 \text{ iff } xy \in \{\alpha \mid \mathsf{abs}(\alpha) \leq N/4\}$$

Combining the above, we conclude that,

$$C_{N,x}(y) = 1 \text{ iff } \alpha_{max} \in \{\alpha \mid \mathsf{abs}(\alpha) \leq N/4\}$$

□

Lemma 6.56.3.

- *If $C_{N,x}(y) = 1$, then $\alpha_{max} \in \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}$ and $\left|\widehat{C_{N,xy}}(\alpha_{max})\right|^2 + \left|\widehat{C_{N,xy}}(-\alpha_{max})\right|^2 \geq 2\tau_{max}$,*
- *If $C_{N,x}(y) = -1$, then $\sum_{\alpha \in \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}} \left|\widehat{C_{N,xy}}(\alpha)\right|^2 \leq 1 - 2\tau_{max}$.*

Proof. Consider first the case that $C_{N,x}(y) = 1$. By Lemma 6.56.2, this implies that $\alpha_{max} \in \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}$, and due to symmetry of $\left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}$ also $-\alpha_{max} \in \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}$. Since $C_{N,xy}$ is a symmetric function,[29] then $\widehat{C_{N,xy}}(\alpha_{max}) = \widehat{C_{N,xy}}(-\alpha_{max})$, implying that

$$\left|\widehat{C_{N,xy}}(\alpha_{max})\right|^2 + \left|\widehat{C_{N,xy}}(-\alpha_{max})\right|^2 = 2\tau_{max}$$

Consider next the case that $C_{N,x}(y) = -1$. By Lemma 6.56.2, this implies that $\alpha_{max} \notin \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}$, and due to symmetry of $\left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}$ also $-\alpha_{max} \notin \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}$. Thus, $\sum_{\alpha \notin \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}} \left|\widehat{C_{N,xy}}(\alpha)\right|^2 \geq 2\tau_{max}$. This implies that

$$\sum_{\alpha \in \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}} \left|\widehat{C_{N,xy}}(\alpha)\right|^2 \leq 1 - 2\tau_{max}$$

where the latter is true since by Parseval Identity $\sum_{\alpha \in \mathbb{Z}_N} \left|\widehat{C_{N,xy}}(\alpha)\right| = \|C_{N,Cy}\|_2^2$ and $\|C_{N,Cy}\|_2^2 = 1$ since $C_{N,Cy}$ accepts $\{\pm 1\}$ values. □

□

Lemma 6.57. *For any $\rho \in (0, 1)$, $\ell = \Theta(\zeta^{1.5}\rho N)$ sufficiently small, and $c \in [0, \lfloor((1-\rho)\frac{N}{4})\rfloor]$, the value $decision_c$ computed in step 2c of the Self Correcting Algorithm satisfies the following with probability at least $1 - \delta$:*

1. *If $\exists \alpha \in \{\alpha \mid \mathsf{abs}(\mathsf{abs}(\alpha) - c) \leq \ell/2\}$ s.t. $\left|\widehat{f}(\alpha)\right|^2 + \left|\widehat{f}(-\alpha)\right|^2 \geq 2\tau_{max} - \varepsilon$, then $decision_c = 1$*
2. *If $\sum_{\alpha \in \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}} \left|\widehat{f}(\alpha)\right|^2 \leq \frac{1}{2}$, then $decision_c = 0$.*

Proof. We first prove the first part of the lemma. Assume there is $\alpha \in \{\alpha \mid \mathsf{abs}(\mathsf{abs}(\alpha) - c) \leq \ell/2\}$ s.t. $\left|\widehat{f}(\alpha)\right|^2 + \left|\widehat{f}(-\alpha)\right|^2 \geq 2\tau_{max} - \varepsilon$. In this case, $\sum_{\alpha\colon \mathsf{abs}(\mathsf{abs}(\alpha)-c) \leq \ell/2} \left|\widehat{f}(\alpha)\right|^2 \geq 2\tau_{max} - \varepsilon$. By Lemma 6.57.1, this implies that $\mathsf{est} \geq 2\tau_{max} - \varepsilon - \zeta = \frac{1}{2} + \zeta$ with probability at least $1 - \delta$ (where the last equality is by assigning the value of ζ). Thus, $decision_c = 1$.

We next prove the second part of the lemma. We show the contra positive, that is, we show that $decision_c = 1$ implies that $\sum_{\alpha \in \left\{-\frac{N}{4}, \ldots, \frac{N}{4}\right\}} \left|\widehat{f}(\alpha)\right|^2 > \frac{1}{2}$. Assume

[29] A function $g\colon \mathbb{Z}_N \to \mathbb{C}$ is *symmetric* if $g(x) = g(-x)$ for all $x \in \mathbb{Z}_N$.

decision$_c = 1$. In this case, $\mathsf{est} > \frac{1}{2} + \zeta$, which by Lemma 6.57.1 with probability at least $1-\delta$ implies that $\sum_{\alpha\colon \mathsf{abs}(\mathsf{abs}(\alpha)-c)\leq \bar{\ell}/2} \left|\widehat{f}(\alpha)\right|^2 > \frac{1}{2}$ for $\bar{\ell} = \sqrt{\frac{128}{3\zeta}}(1+\frac{20}{3\zeta})\ell$. Observing that $\left\{\alpha\colon \mathsf{abs}(\mathsf{abs}(\alpha)-c) \leq \bar{\ell}/2\right\} \subseteq \left\{-\frac{N}{4},\dots,\frac{N}{4}\right\}$ for any $c \leq (1-\rho)\frac{N}{4}$, we conclude that $\sum_{\alpha\in\left\{-\frac{N}{4},\dots,\frac{N}{4}\right\}} \left|\widehat{f}(\alpha)\right|^2 > \frac{1}{2}$.

Lemma 6.57.1. *If $\zeta > 0$, then with probability at least $1-\delta$ the following holds for* $\mathsf{est} = \mathsf{est}_+ + \mathsf{est}_-$*:*

1. $\mathsf{est} \geq \sum_{\alpha\colon \mathsf{abs}(\mathsf{abs}(\alpha)-c)\leq \ell/2} \left|\widehat{f}(\alpha)\right|^2 - \zeta$

2. $\sum_{\alpha\colon \mathsf{abs}(\mathsf{abs}(\alpha)-c)\leq \bar{\ell}/2} \left|\widehat{f}(\alpha)\right|^2 > \mathsf{est} - \zeta$ *for any* $\bar{\ell} \geq \sqrt{\frac{128}{3\zeta}}(1+\frac{20}{3\zeta})\ell$

Proof. Denote $\gamma = \zeta/4$, $\gamma' = \sqrt{\frac{3}{5}\gamma}$ and $\ell' = (1+\frac{1}{\gamma'})\ell$.

We prove the first part of the lemma. By Proposition 6.58, since $\ell = \gamma'\ell'$ and m_A, m_B set in Algorithm 6.53 are of sizes $m_A = \Theta(\frac{1}{\zeta^2}\ln\frac{1}{\delta})$ and $m_B = \Theta(\frac{1}{\zeta^2}\ln\frac{1}{\delta}\ln m_A)$, then, with probability at least $1-\delta$:

$$\mathsf{est}_+ \geq \sum_{\alpha\colon \mathsf{abs}(\alpha-c)\leq \ell/2} \left|\widehat{f}(\alpha)\right|^2 (1-\frac{5}{6}(\gamma')^2) - \gamma$$

Observing that $\sum_{\alpha\colon \mathsf{abs}(\alpha-c)\leq \ell/2} \left|\widehat{f}(\alpha)\right|^2 \leq 1$ and $\frac{5}{6}(\gamma')^2, \gamma \leq \zeta/4$ we conclude that $\mathsf{est}_+ \geq \sum_{\alpha\colon \mathsf{abs}(\alpha-c)\leq \ell/2} \left|\widehat{f}(\alpha)\right|^2 - \zeta/2$. Similarly, $\mathsf{est}_- \geq \sum_{\alpha\colon \mathsf{abs}(\alpha+c)\leq \ell/2} \left|\widehat{f}(\alpha)\right|^2 - \zeta/2$. Combining these two bounds concludes the proof of the first part, as it shows that $\mathsf{est} = \mathsf{est}_+ + \mathsf{est}_-$ satisfies that

$$\mathsf{est} \geq \sum_{\alpha\colon \mathsf{abs}(\mathsf{abs}(\alpha)-c)\leq \ell/2} \left|\widehat{f}(\alpha)\right|^2 - \zeta$$

We prove the second part of the lemma. By Proposition 6.58, since $\bar{\ell} \geq \sqrt{\frac{32}{3\gamma}}\ell$ and m_A, m_B set in Algorithm 6.53 are of sizes $m_A = \Theta(\frac{1}{\zeta^2}\ln\frac{1}{\delta})$ and $m_B = \Theta(\frac{1}{\zeta^2}\ln\frac{1}{\delta}\ln m_A)$, then, with probability at least $1-\delta$:

$$\sum_{\alpha\colon \mathsf{abs}(\alpha-c)\leq \bar{\ell}/2} \left|\widehat{f}(\alpha)\right|^2 > \mathsf{est}_+ - 2\gamma$$

Similarly, $\sum_{\alpha\colon \mathsf{abs}(\alpha+c)\leq \bar{\ell}/2} \left|\widehat{f}(\alpha)\right|^2 > \mathsf{est}_- - 2\gamma$. Combining these two bounds and assigning the value $\gamma = \zeta/4$, we get that

$$\sum_{\alpha\colon \mathsf{abs}(\mathsf{abs}(\alpha)-c)\leq \bar{\ell}/2} \left|\widehat{f}(\alpha)\right|^2 > \mathsf{est} - 4\gamma = \mathsf{est} - \zeta$$

□

Proposition 6.58. *For any $N \in \mathbb{N}$, $\gamma, \delta \in (0,1)$, $c \in \mathbb{Z}_N$, $\ell' \leq N/2$ and $f\colon \mathbb{Z}_N \to \{\pm 1\}$, properties 1-2 below are satisfied with probability at least $1-\delta$ by the value*

$$\mathsf{estimator} = \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B|}\sum_{y\in B}\chi_{-c}(y)f(x-y)\right)^2$$

for $A \subseteq \mathbb{Z}_N$ and $B \subseteq \left\{0,\dots,\frac{N}{2\ell'}-1\right\}$ subsets of sizes $m_A = \Theta(\frac{1}{\gamma^2}\ln\frac{1}{\delta})$ and $m_B = \Theta(\frac{1}{\gamma^2}\ln\frac{1}{\delta}\ln m_A)$ chosen uniformly at random.

1. *For any $\gamma' \in (0,1]$, $\mathsf{estimator} \geq \sum_{\alpha\colon \mathsf{abs}(\alpha-c)\leq\gamma'\frac{\ell'}{2}}\left|\widehat{f}(\alpha)\right|^2(1-\frac{5}{6}(\gamma')^2)-\gamma$*
2. *$\sum_{\alpha\colon \mathsf{abs}(\alpha-c)\leq\sqrt{\frac{32}{3\gamma}}\frac{\ell'}{2}}\left|\widehat{f}(\alpha)\right|^2 > \mathsf{estimator}-2\gamma$*

Proof. This proposition follows from combining Lemma 3.20 with Lemma 3.19 (when using its strong form as stated in the remark following it). □

6.6 Linear Encoding Length via Restrictions: Proof of Theorem 6.34

We prove Theorem 6.34 showing there exists codes for finite abelian groups of linear encoding length, constant distance and constant list decoding bound.

The codes of linear encoding length are constructed by taking *random restrictions* $\mathcal{C}(S')$ of codes $\mathcal{C}$ for G achieving exponential encoding length, constant distance and constant list decoding bound. That is, codewords of $\mathcal{C}(S')$ are restrictions of the codewords of $\mathcal{C}$ to a subset S' of their entries. The code $\mathcal{C}(S')$ has the same alphabet size as the code $\mathcal{C}$, and has linear encoding length. In Lemma 6.36 we show that the code $\mathcal{C}(S')$ maintains roughly the same distance as the code $\mathcal{C}$ with high probability. We thus obtain codes of linear encoding length and constant distance and list decoding bound from the codes of exponential encoding length, and constant distance and list decoding bound.

Definition 6.35. [Restrictions] For any code $\mathcal{C} \subseteq \Sigma^S$ with entries indexed by elements in S and a subset $S' \subseteq S$, we defined the *restriction of $\mathcal{C}$ to S'* to be the code $\mathcal{C}(S') = \left\{C_{|S}\right\}_{C\in\mathcal{C}}$ for $C_{|S} \in \Sigma^S$ the restriction of C to entries whose index s is in S'. We say that $\mathcal{C}(S')$ is an *n-random restriction of $\mathcal{C}$* if the set S' is a a subset of S of size n chosen uniformly at random.

Example 6.59. *Let $\mathcal{C}^{const} = \mathcal{C}^{const}(\gamma, N) = (\mathbb{Z}_N, \mathsf{half}_N, S')$ be the restriction of the code $(\mathbb{Z}_N, \mathsf{half}_N, \mathbb{Z}_N)$ from Example 6.2 to a random subset $S' \subseteq \mathbb{Z}_N$ of size $O(\log N/\gamma^2)$. Then $\mathcal{C}^{const}$ is a binary code of constant rate, and with high probability it has constant distance (where the probability is taken over the choice of S').*

Theorem 6.34. For any $\gamma < 0.17$ and any finite abelian group G, there exists Multiplication codes $\mathcal{C}$ for G of alphabet Σ, rate $O(1/\gamma^2 \log |\Sigma|)$, constant distance, and satisfying the following list decoding bound: $\forall C \in \mathcal{C}$, $\left|Ball_{\mathcal{C}}(C, \frac{1}{2} - 2\gamma)\right| \leq poly(\log |G|, 1/\gamma)$.[30]. The alphabet size $|\Sigma|$ depends on the underlying group: $|\Sigma| = 2$ for cyclic groups, $|\Sigma| = O(1)$ for groups with constant size generating set, and $|\Sigma| = |G|$ for arbitrary finite abelian groups. Algebraic operations in these decoding algorithms are computed in the group G.

Proof. Let $\mathcal{C}$ be a code for G achieving exponential encoding length, constant distance and constant list decoding bound. Let $\mathcal{C}(S')$ be a random restrictions of $\mathcal{C}$ to a set of size $O(\log |G| / \gamma^2)$. The code $\mathcal{C}(S')$ has linear encoding length and the same alphabet size as the code $\mathcal{C}$. In Lemma 6.36 we show that the distance of the code $\mathcal{C}(S')$ is at least $\delta - \gamma$ for δ the distance of the code $\mathcal{C}$. We thus obtain codes of linear encoding length and constant distance and list decoding bound from the codes of exponential encoding length, and constant distance and list decoding bound. $\square$

Lemma 6.36. Let $\mathcal{C} \subseteq \Sigma^S$ be a code of non-linear encoding length $|S| = \omega(\log |\mathcal{C}|)$ and with relative Hamming distance at least δ. For any $\gamma > 0$ and any $n = \Theta(\log N/\gamma^2)$ sufficiently large, the n-random restriction of $\mathcal{C}$ has linear encoding length of $O(1/\gamma^2)$ and its relative Hamming distance is at least $\delta - \gamma$ with probability at least $1 - O(\frac{1}{|\mathcal{C}|})$ (where the probability is taken over the choice of the random restriction $S' \subseteq S$).

Proof. We show that for any two codewords C, C' of $\mathcal{C}$ and the corresponding two codewords $C_{|S'}, C'_{|S'}$ of $\mathcal{C}(S')$, the distance $\|C_{|S'} - C'_{S'}\|_2^2$ of $C_{|S'}, C'_{|S'}$ is at least $\|C - C'\|_2^2$ with high probability.

Recall that $\Delta(C_{|S'}, C'_{|S'}) = \Pr_{s \in S'}[C(s) \neq C'(s)]$. By Chernoff bound

$$\Pr_{S'}\left[\left|\Delta(C_{|S'}, C'_{|S'}) - \Delta(C, C')\right| > \gamma\right] < 2\exp(-2|S'|\gamma^2)$$

By union bound, this holds for every $C, C' \in \mathcal{C}$ with probability at least
$1 - 2|\mathcal{C}|^2 \exp(-2|S'|\gamma^2) = 1 - O(\frac{1}{|\mathcal{C}|})$ where the last equality is by choice of $|S'| = \Theta(\frac{\log|\mathcal{C}|}{\gamma^2})$ large enough. $\square$

6.7 Codes for Groups of Small Generating Sets: Proof of Theorem 6.32

We sketch the decoding and self correcting algorithms for the codes for $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ in Definition 6.31 and their analysis, proving the derivation rules stated in Theorem 6.32.

[30] $Ball_{\mathcal{C}}(C, r)$ denotes the set of codewords of $\mathcal{C}$ within relative Hamming distance at most r from C.

Theorem 6.32. For $i = 1, \ldots, k$, let $\mathcal{C}_{\mathbb{Z}_{N_i}}$ be codes for the groups $\mathbb{Z}_{N_i}$ of alphabet Σ, rate r, relative Hamming distance Δ, which are $(O(1), \ell, q, T)$-decodable and $(O(1), q, T, 1-\rho)$-locally self correctable. Let $\mathcal{C}_G$ be the code for the group $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ built from the code $\mathcal{C}_{\mathbb{Z}_N}$ as in Definition 6.31. Then the code $\mathcal{C}_G$ is a code for the group $\mathbb{Z}_N^k$ of alphabet Σ^k, rate r, relative Hamming distance $1-O(1)$, which is $(\frac{1}{2}-O(1), 1, q, kT)$-decodable and $(\frac{1}{2}-O(1), q, kT, (1-\rho)^k)$-locally self correctable.

Proof. We sketch the decoding and self correcting algorithms for the codes for $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ in Definition 6.31 and their analysis.

At a high level both these algorithms operates by first decomposing the given corrupted codewords C'_m into k corrupted codewords $C'_{m_1}, \ldots, C'_{m_k}$, such that if C'_m is close to the codeword encoding message $m = (m_1, \ldots, m_k)$ in the code for $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$. then each C'_{m_i} is close to the codeword C_{m_i} encoding the message m_i in the code for $\mathbb{Z}_{N_i}$. Then they apply the decoding/self correcting algorithms for codes for $\mathbb{Z}_N$ on the corrupted codewords C'_{m_i}. finally they combine the values to the output.

Elaborating on the decomposition step, query access for the corrupted codeword C'_{m_i} is gained as follows. Given $i \in [k]$ and entry location ℓ, output the value computed as follows. Randomly sample $O_\varepsilon(1)$ of the neighbors of left node ℓ in the graph H_i. Parse the corresponding entries in the corrupted codeword C'_m to find the values corresponding to the encoding of m_i. Let the ℓ-th entry of C'_{m_i} be the majority vote over all these values.[31] It is known [44] that the majority over all neighbors yields the correct value with probability $1 - O_d(1)$. By Chernoff bound, with high probability, the value is correct with probability $1 - O_d(1)$, even when the majority vote is only over a random subset of the neighbors as done here. Thus, C'_{m_i} is at relative Hamming distance $O_d(1)$ from the codeword C_{m_i} encoding m_i with high probability.

Next we elaborate on the decoding/self correcting step for C'_{m_i} and the combining step. The decoding algorithm uses as building-block decoding algorithms $A_1, \ldots, A_k$ for the codes $\mathcal{C}_{N_1}, \ldots, \mathcal{C}_{N_k}$. For each $i \in [k]$, it applies the decoding algorithm A_i on the corrupted codeword C'_{m_i} to find messages m'_i, and outputs $m' = (m'_1, \ldots, m'_k)$. By properties of the decoding algorithms A_i, for each i, $m'_i = m_i$ with probability at least $2/3$. Thus the output m' is the encoded message m with probability at least $(2/3)^k$. The running time of this algorithm is $O(d\sum_{i=1}^k T_i)$ for T_i the running time of A_i, and the number of queries it makes is $O(d\sum_{i=1}^k q_i)$ for q_i the number of queries made by A_i.

Similarly, the self correcting algorithm uses as building-block self correcting algorithms $A'_1, \ldots, A'_k$ for the codes $\mathcal{C}_{N_1}, \ldots, \mathcal{C}_{N_k}$. For each $i \in [k]$, and each $j \in [d]$, it applies the self correcting algorithm A_i on entry location corresponding to the j-th

[31]This algorithm is an adaptation of the Guruswami-Indyk [45] majority decoding algorithm for ABNNR codes: In the Guruswami-Indyk [45] algorithm, the value of entry ℓ is determined by the majority vote over *all* neighbors. In our settings, the number of neighbors may be huge. For example, consider the group $\mathbb{Z}_2 \times \mathbb{Z}_N$ and the case $n = N$. In this case, the number of neighbors in each left node in the graph $H_1 = ([2], [n], E_1)$ is $Nd/2$. So the actual majority vote cannot be decided in time $poly(\log N)$. Nevertheless, the majority vote over a sample of a random subset of the neighbors gives the value of the true majority with high probability, and can be computed efficiently.

neighbor of right node s in the graph H^i and query access to the corrupted codeword C'_{m_i} to find values of C_{m_i} on entries corresponding to the d neighbors of right node s, denote these symbols by $b_{i1}, \ldots, b_{id}$. The output is $((b_{11}, \ldots, b_{k1}), \ldots, ((b_{1d}, \ldots, b_{kd})$. By properties of the self correcting algorithm A'_i, each symbol b_{ij} is the values of C_{m_i} on entries corresponding to the j-th neighbor of right node s in the graph H^i, with probability at least $1-\rho$. Thus by definition of the code for G, the output is the value of the s entry in C_m, with probability at least $(1-\rho)^{kd}$. The running time of this algorithm is $O(d\sum_{i=1}^k T_i)$ for T_i the running time of A_i, and the number of queries it makes is $O(d\sum_{i=1}^k q_i)$ for q_i the number of queries made by A_i. $\square$

6.8 Soft Error Reduction for Concatenated ABNNR Codes: Proof of Theorem 6.43

In this section we present our soft error reduction algorithm for concatenated ABNNR codes –with possibly unbalanced expander graphs– and analyze its correctness and running time complexity.

Definition 6.60 (unbalanced expander)**.** *For unbalanced bipartite graphs $H = ([k], [b], E)$, we say that H is a* λ-expander *if for every $A \subseteq [k]$ and $B \subseteq [b]$,*

$$\left| |E(A,B)| - \frac{d_L\,|A|\,|B|}{b} \right| < \lambda d_L \sqrt{|A|\,|B|}$$

Remark 6.61. *A balanced d-regular λ-expander graph $H = ([n], [n], E)$ satisfies the above property, namely, for every $A \subseteq [k]$ and $B \subseteq [b]$, $\left||E(A,B)| - \frac{d|A||B|}{b}\right| < \lambda d\sqrt{|A|\,|B|}$. This is bound is known as the* Expander Mixing Lemma.

Theorem 6.62 (Soft Error Reduction)**.** *Let $\mathcal{C}(\mathcal{H}, \mathcal{C}_0)$ be a binary ABNNR code with $\mathcal{H}$ a family of (d_L, d_R)-regular λ-expander graphs $H = ([k], [b], E)$, and $\mathcal{C}_0$ a family of binary error correcting codes of rate r and relative Hamming distance $\Delta(\mathcal{C}_0)$.*

There is a soft error reduction algorithm that for each $k \in \mathbb{N}$, a message $x \in \{0,1\}^k$, a noise parameter $\varepsilon \in [0, 1-2\Delta(\mathcal{C}_0))$ and a recovery parameter $\gamma \in (O(\lambda^{1/3}), O(\varepsilon))$,[32] *given $w \in \{0,1\}^{k/r}$ s.t. $\Delta(w, C(x)) \leq \frac{1}{4} - \varepsilon$ for $C(x) \in \mathcal{C}(\mathcal{H}, \mathcal{C}_0)$ the encoding of x in the binary ABNNR code, outputs $z \in \{0,1\}^k$ s.t.*

$$\Delta(x, z) \leq \gamma$$

The running time of the algorithm in $O\left(k \cdot 2^{d_L d_R}\right)$.

Proof. Fix an input corrupted codeword w and let x be the encoded message s.t. $\Delta(C(x), w) < \frac{1}{4} - \varepsilon$. Recall that for each left node ℓ, we denote by $dist_\ell(y) =$

[32] Specifically, we require $\gamma \in ((\sqrt{2}\lambda)^{1/3}, \frac{3}{8}\varepsilon)$

$\frac{1}{d}\sum_{i\in\Gamma(\ell)}\Delta(C_0(y)^i, w^i)$ the average distance of codewords encoding (messages agreeing with) y from w on the block corresponding to right neighbors i of left node ℓ.

Algorithm 6.63. Soft Error Reduction Algorithm.
Input: *A description of the code $\mathcal{C}(\mathcal{H},\mathcal{C}_0)$, a noise parameter ε and a corrupted codeword $w \in \{0,1\}^{b/r}$ s.t. $\Delta(C(x),w) \leq \frac{1}{4}-\frac{\varepsilon}{4}$.*
Output: *$z \in \{0,1\}^n$ s.t. $\Delta(z,x) \leq \gamma$.*
Steps:

```
1. For each $\ell = 1,\ldots,k$
    (a) By exhaustive search, find assignment $y \in \{0,1\}^{d_R d_L}$ to all message bits
        $i \in \Gamma_L(\Gamma_R(\ell))$ such that $y$ minimizes
            $dist_\ell(y) = \frac{1}{d_L}\sum_{i\in\Gamma(\ell)}\Delta(C_0(y)^i, w^i)$
    (b) If $dist_\ell(y) < \frac{1}{4}-\frac{\varepsilon}{4}$, set $z_\ell = y_\ell$
    (c) Else set $z_\ell = \perp$
2. Output $z_1 \ldots z_k$
```

By Lemma 6.64 below when assigning $\gamma \leq \varepsilon/8$, for at least $(1-\gamma)n$ of the bits $\ell \in [k]$ of the encoded message x,

$$dist_\ell(x) < \frac{1}{4}-\frac{\varepsilon}{4}$$

Conversely, by Lemma 6.65 below, for each left node $\ell \in [k]$ and any $y \in \{0,1\}^k$ such that $y_\ell \neq x_\ell$,

$$dist_\ell(y) > \frac{1}{4}-\frac{\varepsilon}{4}$$

Combinning both lemmata together, we conclude that for at least $(1-\gamma)k$ of the bits $\ell \in [k]$, $z_\ell = x_\ell$ (for z_ℓ from Algorithm 6.42 above).

The running time of this algorithm is $k \cdot 2^{d^2}$. □

We now prove the lemmata used in the proof of Theorem 6.62 above. We use the following notation: For each right node $i \in [b]$ and any assignment $y \in \{0,1\}^{d_R}$ to the left neighbors of i, we denote the distance of the i-th block of w from $C_0(y)$ (*i.e.*, the i-th block of the encoding of any message agreeing with y) by

$$\Delta_i(y) \overset{def}{=} \Delta(C_0(y)^i, w^i)$$

We keep the same notation as in Theorem 6.62.

Lemma 6.64 (Main Lemma)**.** *For any $\gamma \in (0, 1/3)$, if $\lambda < \gamma^3/\sqrt{2}$, then for any $x \in \{0,1\}^k$, for at least $(1-\gamma)k$ of the bits $\ell \in 1, \ldots, k$,*

$$\left| \mathop{\mathbb{E}}_{i\in\Gamma(\ell)} [\Delta_i(x)] - \mathop{\mathbb{E}}_{i\in 1,\ldots,b} [\Delta_i(x)] \right| \leq 2\gamma$$

Proof. We first fix some notation. Denote $\delta = \gamma^2 d_L$. Fix some $x \in \{0,1\}^k$. For $j = 1, \ldots, 1/\gamma$, denote

$$T_j = \{ i \in 1, \ldots, b \mid \Delta_i(x) \in [(j-1)\gamma, j\gamma) \}$$

Claim 6.64.1. *For any $\ell \in 1, \ldots, k$, if $\forall j \in 1, \ldots, 1/\gamma$, $\left| E(\{\ell\}, T_j) - d_L \frac{|T_j|}{b} \right| \leq \delta$, then*

$$\left| \mathop{\mathbb{E}}_{i\in\Gamma(\ell)} [\Delta_i(x)] - \mathop{\mathbb{E}}_{i\in 1,\ldots,b} [\Delta_i(x)] \right| \leq 2\gamma$$

Proof. First we bound $\mathbb{E}_{i\in\Gamma(\ell)}[\Delta_i(x)]$. Rewriting $\mathbb{E}_{i\in\Gamma(\ell)}[\Delta_i(x)]$ in terms of the T_j's we get

$$\mathop{\mathbb{E}}_{i\in\Gamma(\ell)} [\Delta_i(x)] \;=\; \frac{1}{|\Gamma(\ell)|} \sum_{i\in\Gamma(\ell)} \Delta_i = \frac{1}{d_L} \sum_{j=1}^{1/\gamma} \sum_{i\in T_j \cap \Gamma(\ell)} \Delta_i$$

(where in the last step we use the fact that $|\Gamma(\ell)| = d_L$, because H is (d_L, d_R)-regular). By the definition of T_j, for all $i \in T_j$, $\Delta_i \in [(j-1)\gamma, j)$ for all $i \in T_j$. Therefore,

$$\frac{1}{d_L} \sum_{j=1}^{1/\gamma} |T_j \cap \Gamma(\ell)| \, (j-1)\gamma \;\leq\; \mathop{\mathbb{E}}_{i\in\Gamma(\ell)} [\Delta_i(x)] < \frac{1}{d_L} \sum_{j=1}^{1/\gamma} |T_j \cap \Gamma(\ell)| \, j\gamma$$

Now, $|T_j \cap \Gamma(\ell)| = |E(\{\ell\}, T_j)| \in d_L \frac{|T_j|}{b} \pm \delta$, therefore,

$$\frac{1}{d_L} \sum_{j=1}^{1/\gamma} \left(d_L \frac{|T_j|}{b} + \delta \right) (j-1)\gamma \;\leq\; \mathop{\mathbb{E}}_{i\in\Gamma(\ell)} [\Delta_i(x)] < \frac{1}{d_L} \sum_{j=1}^{1/\gamma} \left(d_L \frac{|T_j|}{b} - \delta \right) j\gamma$$

Canceling the d_L in the denominator and the nominator we get

$$\sum_{j=1}^{1/\gamma} \left(\frac{|T_j|}{b} - \frac{\delta}{d_L} \right) (j-1)\gamma \;\leq\; \mathop{\mathbb{E}}_{i\in\Gamma(\ell)} [\Delta_i(x)] < \sum_{j=1}^{1/\gamma} \left(\frac{|T_j|}{b} + \frac{\delta}{d_L} \right) j\gamma \qquad (6.1)$$

Second, we bound $\mathbb{E}_{i=1,\ldots,b}[\Delta_i(x)]$. Rewriting $\mathbb{E}_{i=1,\ldots,b}[\Delta_i(x)]$ in terms of the T_j's we get

$$\mathop{\mathbb{E}}_{i=1,\ldots,b} [\Delta_i(x)] = \frac{1}{b} \sum_{i=1}^{b} \Delta_i = \frac{1}{b} \sum_{j=1}^{1/\gamma} \sum_{i\in T_j} \Delta_i$$

By the definition of T_j, for all $i \in T_j$, $\Delta_i \in [(j-1)\gamma, j)$ for all $i \in T_j$. Therefore,

$$\sum_{j=1}^{1/\gamma} \frac{|T_j|}{b}(j-1)\gamma \;\; \leq \;\; \mathop{\mathbb{E}}_{i=1,\ldots,b}[\Delta_i(x)] < \sum_{j=1}^{1/\gamma} \frac{|T_j|}{b} j\gamma \tag{6.2}$$

Combining the bounds from Equations 6.1 and 6.2 we conclude that

$$\left| \mathop{\mathbb{E}}_{i\in\Gamma(\ell)}[\Delta_i(x)] - \mathop{\mathbb{E}}_{i=1,\ldots,b}[\Delta_i(x)] \right| \leq \sum_{j=1}^{1/\gamma} \frac{|T_j|}{b}\gamma + \sum_{j=1}^{1/\gamma} \frac{\delta}{d_L} j\gamma \leq 2\gamma$$

(where the last inequality is true since $\sum_{j=1}^{1/\gamma} \frac{|T_j|}{b}\gamma = \gamma$, and $\sum_{j=1}^{1/\gamma} \frac{\delta}{d_L} j\gamma = \frac{\delta}{d_L}\gamma \frac{\frac{1}{\gamma}(\frac{1}{\gamma}+1)}{2} < \frac{\delta}{\gamma d_L} = \gamma$ for any $\gamma < 1/3$ and $\delta = \gamma^2 d_L$). □

Claim 6.64.2. *Denote* $Bad = \left\{ \ell \mid \exists j \in 1,\ldots,1/\gamma \text{ s.t. } \left| E(\{\ell\},T_j) - d_L \frac{|T_j|}{b} \right| > \delta \right\}$. *If* $\lambda < \gamma^3/\sqrt{2}$, *then* $|Bad| < \gamma b$

Proof. For each $j \in 1,\ldots,\gamma$, denote

$$\begin{aligned} Bad_j^+ &= \left\{ \ell \mid |E(\{\ell\},T_j)| > d_L \frac{|T_j|}{b} + \delta \right\} \\ Bad_j^- &= \left\{ \ell \mid |E(\{\ell\},T_j)| < d_L \frac{|T_j|}{b} - \delta \right\} \end{aligned}$$

By a counting argument, there exists $j \in 1,\ldots,1/\gamma$ such that either $\left|Bad_j^+\right| \geq \frac{\gamma}{2}|Bad|$ or $\left|Bad_j^-\right| \geq \frac{\gamma}{2}|Bad|$. Without loss of generality assume

$$\left|Bad_j^+\right| \;\; \geq \;\; \frac{\gamma}{2}|Bad| \tag{6.3}$$

By definition of Bad_j^+, for each $\ell \in Bad_j^+$, $|E(\{\ell\},T_j)| > d_L\frac{|T_j|}{b} + \delta$. Therefore,

$$\left|E(Bad_j^+,T_j)\right| \;\; > \;\; \left|Bad_j^+\right| \left(d_L \frac{|T_j|}{b} + \delta \right) \tag{6.4}$$

On the other hand, by Expander Mixing Lemma ,

$$\left|E(Bad_j^+,T_j)\right| \;\; \leq \;\; d_L \left|Bad_j^+\right| \frac{|T_j|}{b} + \lambda d_L \sqrt{\left|Bad_j^+\right| |T_j|} \tag{6.5}$$

Combining Equations 6.4 and 6.5, we get that

$$\left|Bad_j^+\right| \left(d_L \frac{|T_j|}{b} + \delta \right) \;\; < \;\; d_L \left|Bad_j^+\right| \frac{|T_j|}{b} + \lambda d_L \sqrt{\left|Bad_j^+\right| |T_j|}$$

Reorganizing the above expression and assigning $\delta = \gamma^2 d_L$ and $|T_j| \leq b$, we get

$$\sqrt{|Bad_j^+|} \;<\; \frac{\lambda d_L}{\delta}\sqrt{|T_j|} \leq \frac{\lambda}{\gamma^2}\sqrt{b}$$

Combining the above with Equation 6.3, we conclude that

$$|Bad| \;<\; \frac{2}{\gamma}\left(\frac{\lambda}{\gamma^2}\right)^2 b < \gamma b$$

(where the last inequality is true by the condition of λ). □

□

Lemma 6.65. *For any $x, y \in \{0,1\}^k$ s.t. $x_\ell \neq y_\ell$, if $\mathbb{E}_{i\in\Gamma(\ell)}[\Delta_i(x)] < \frac{1}{4} - \frac{\varepsilon}{4}$ then*

$$\mathop{\mathbb{E}}_{i\in\Gamma(\ell)}[\Delta_i(y)] > \frac{1}{4} - \frac{\varepsilon}{4}$$

Proof. Recall that x^i, y^i denote the restrictions of x and x, y, respectively, to the bits $i_1, \ldots, i_{d_R} \in 1, \ldots, k$ neighboring the i-th right node of H. For all $i \in \Gamma(\ell)$, $x^i \neq y^i$, because ℓ is a neighbor of each $i \in \Gamma(\ell)$, *i.e.*, $\ell \in i_1, \ldots, i_{d_R}$. Therefore,

$$\forall i \in \Gamma(\ell),\ \Delta(x^i, y^i) \geq dist(C_0) \geq \frac{1}{2} - \frac{\varepsilon}{2} \tag{6.6}$$

By the triangle inequality,

$$\Delta(x^i, y^i) \leq \Delta(x^i, w^i) + \Delta(w^i, y^i) \tag{6.7}$$

Combining Equations 6.6 and 6.7 above, we get that

$$\Delta(w^i, y^i) \geq \Delta(x^i, y^i) - \Delta(x^i, w^i) \geq \frac{1}{2} - \frac{\varepsilon}{2} - \Delta(x^i, w^i)$$

Taking the expectation over all $i \in \Gamma(\ell)$, we conclude that

$$\mathop{\mathbb{E}}_{i\in\Gamma(\ell)}\left[\Delta(w^i, y^i)\right] \geq \frac{1}{2} - \frac{\varepsilon}{2} - \mathop{\mathbb{E}}_{i\in\Gamma(\ell)}\left[\Delta(x^i, w^i)\right] \geq \frac{1}{2} - \frac{\varepsilon}{2} - \left(\frac{1}{4} - \frac{\varepsilon}{4}\right) = \frac{1}{4} - \frac{\varepsilon}{4}$$

□

Chapter 7

Cryptographic Hardcore Predicates via List Decoding

We introduce a unifying framework for proving that predicate P is hardcore for a one-way function f, and apply it to a broad family of functions and predicates, showing new hardcore predicates for well known one-way function candidates as well as reproving old results in an entirely different way.

Our framework extends the list-decoding method of Goldreich and Levin for showing hardcore predicates. Namely, a predicate will correspond to some error correcting code, predicting a predicate will correspond to access to a corrupted codeword, and the task of inverting one-way functions will correspond to the task of list decoding a corrupted codeword.

A characteristic of the error correcting codes which emerge and are addressed by our framework, is that codewords can be approximated by a small number of heavy coefficients in their Fourier representation. Moreover, as long as corrupted words are close enough to legal codewords, they will share a heavy Fourier coefficient. We list decode such codes, by devising a learning algorithm applied to corrupted codewords for learning heavy Fourier coefficients. Details on this learning algorithm are given in chapter 3.

Our algorithm for learning heavy Fourier coefficients is applicable when query access to the corrupted codeword is given. We obtain such query access whenever the considered one-way function f satisfies that given $f(x)$ and y one can efficiently compute $f(xy)$. This is the case, for example, for the well known one-way function candidates of RSA, Rabin, Discrete Logarithm modulo primes, and Discrete Logarithm in elliptic curves.

For the Diffie-Hellman (DH) candidate one-way function we reduce the problem of proving hardcore predicates for DH to the problem of learning heavy Fourier coefficients (LCN) in various weak access models such as the random samples access model. No learning algorithm in these access models is known. This reduction can be interpreted either as a direction for proving hardcore predicates for DH, or as evidence toward the intractability of LCN under those access models.

7.1 Introduction

Let f be a one-way function, namely a function which is easy to compute, but hard to invert on all but a negligible fraction of its inputs. We say that a Boolean predicate P is a *hardcore predicate for* f if $P(x)$ is easy to compute given x, but hard to guess with non-negligible advantage beyond 50% given only $f(x)$. The notion of hardcore predicates was introduced and investigated in [18, 41] and has since proven central to cryptography and pseudo-randomness.

The standard proof methodology for showing P is hardcore for f is by a reduction from inverting f to predicting P. That is, demonstrate an efficient inversion algorithm for f, given access to a probabilistic polynomial time magic-algorithm B that on input $f(x)$ guesses $P(x)$ with non-negligible advantage over a random guess. Since f is assumed to be a one-way function, it follows that no such algorithm B exists and P is a hardcore predicate.

Blum and Micali [18] were the first to show a hardcore predicates for a function widely conjectured to be one-way. Let p be a prime and g a generator for Z_p^*. The function $EXP_{p,g}\colon Z_{p-1} \to Z_p^*$, $EXP_{p,g}(x) = g^x \bmod p$ is easy to compute and as hard to invert as solving discrete logarithm modulo a prime p. Blum and Micali [18] define the predicate $BM_{p,g}(x) = 0$ if $0 \leq x < \frac{p-1}{2}$ and 1 otherwise, and prove it is a hardcore predicate for $EXP_{p,g}$ if the discrete logarithm problem is intractable. In subsequent years, it was shown for other conjectured one-way functions f and other predicates P, that P is a hardcore predicates for f [5, 16, 32, 41, 42, 51, 59, 86, 90]. Most notably, for the RSA [75] function $RSA\colon Z_n^* \to Z_n^*$, $RSA(x) = x^e \bmod n$, the predicates $P_i(x) = i^{th}$ bit of x were shown hardcore, first for $i = 1, |n|$ [5] and recently for any $1 \leq i \leq |n|$ [51].

Goldreich and Levin [38] address the general question of whether every one-way function (OWF) has some hardcore predicate. They show that for any OWF $f\colon \{0,1\}^n \to \{0,1\}^*$ one can define another OWF $f'\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^* \times \{0,1\}^n$ by $f'(x,r) = (f(x), r)$, so that the predicate $GL(x,r) = \sum_{i=1}^n x_i r_i$ is a hardcore predicates for f'.

The work of Goldreich-Levin, which explicitly addressed hardcore predicates for arbitrary one-way functions, by way of solution gave a polynomial time list-decoding algorithm for a well known error correcting code – the Hadamard code. It introduced an interesting connection between hardcore predicatess and list decoding which, as pointed out by Impagliazzo and Sudan [56, 82, 87], could potentially lead to a general *list decoding methodology* for proving hardcore predicates for one-way functions.

We formalize such a methodology. Given a function f and predicate P, one would have to:

1. **Define a Code.** Identify an error-correcting code $\mathcal{C}^P$ encoding distinct x's, such that given only $f(x)$ and the capability to compute $P(z)$ on input z, one can query-access the codeword for x, C_x^P. In the case of [38] the code defined was the Hadamard code which is a natural choice as $GL(x,r)$ is precisely the r-th entry of the Hadamard encoding of string x.

2. **List Decode.** Show a polynomial-time list-decoding algorithm for

the code $\mathcal{C}^P$, which works with query access to a corrupted codeword and tolerates a $< \frac{1}{2} - \varepsilon$ fraction of error. In the case of Hadamard code, [38] indeed provides such a list decoding algorithm.

3. Show that predicting P implies access to a corrupted codeword. Show that if there exists a magic algorithm B that on input $f(x)$ predicts $P(x)$ with non-negligible advantage, then there exists an access algorithm which for a non-negligible fraction of x's, on input $f(x)$, can query access a corrupted codeword of x with $< \frac{1}{2} - \varepsilon$ fraction of errors.

Putting all these together, implies that if B exists then f can be inverted for a non-negligible fraction of x's (using the list decoding algorithm). Thus, under the assumption that f is a one-way function, no such B can exist and predicate P is a hardcore predicate.

This is no doubt an elegant methodology but is it a **useful** methodology for proving hardcore predicate results for natural f's and P's? At the very least, can we define appropriate codes and corresponding list decoding algorithms so as to employ this methodology for proving existing hardcore predicate results of [18] for the EXP function, and the results of of [5, 51] for the RSA function?

These questions are the starting point for our investigation.

Our Work

We introduce a unifying framework for proving that predicate P is hardcore for a one-way function f, and apply it to a broad family of functions and predicates, showing new hardcore predicates for well known one-way function candidates as well as reproving old results in an entirely different way.

Our framework follows a list decoding methodology. Thus, the technical essence of the new proofs is to define appropriate codes and to list decode them. These two tasks are independent of the one-way function in question and depend only on the predicate. The only consideration given to the one-way function is in devising a way to access the corrupted codeword.

The error correcting codes which emerge and are addressed by our framework are our Multiplication codes studied in Chapter 6. In these codes, codewords can be approximated by considering only a small number of heavy coefficients in their Fourier representation. Moreover, as long as corrupted codewords are close enough to legal codewords, they will share a heavy Fourier coefficient. To list decode, we use our SFT Algorithm 3.4 (see chapter 3) applied to corrupted codewords for finding their heavy Fourier coefficients, and then find all codewords for which these coefficients are heavy.

Let us now elaborate on the hardcore predicates and one-way functions for which we apply our framework.

Segment Predicates: A Class of New Hardcore Predicates

We apply our framework to prove that a wide class of predicates called *segment predicates* are hardcore predicates for various well known candidate one-way functions.

A segment predicate is any arbitrary assignment of Boolean values to an arbitrary partition of $\mathbb{Z}_N$ into $poly(\log N)$ segments, or a multiplicative shift of such an assignment. A segment predicate can be balanced (with the same number of 0's and 1's) or unbalanced as long as it is far from a constant function. In the latter case of unbalanced predicates, we naturally adapt the definition of hardcore unbalanced predicates to be that it is impossible to compute the predicate better than guessing it at random from $f(x)$.

We prove that any segment predicate is hardcore for any one-way function f defined over $\mathbb{Z}_N$ for which, for a non-negligible fraction of the x's, given $f(x)$ and y, one can efficiently compute $f(xy)$ (where xy is multiplication in $\mathbb{Z}_N$). This includes the functions $EXP_{p,g}$, $RSA(x)$, $Rabin(x) = x^2 \bmod n$, and $ECL_{a,b,p,Q} = xQ$ where Q is a point of high order on an elliptic curve $E_{a,b,p,Q}(Z_p)$ (naturally the appropriate N in each case differs).

In particular, this implies that for every i the *i-partition-bit* is a hardcore predicate for the RSA function where we define the i-th partition bit of x as 0 if $0 \leq 2^i x \leq \frac{N}{2} \bmod N$ and 1 otherwise.

In contrast with the notion of segment predicates, we remark that in the past most all predicates investigated correspond in a fairly direct way with bits in the inverse of $f(x)$. An exception is the work of [69] showing that $P_i(x) = ith$ bit of $ax + b \bmod p$ for randomly chosen a, b, p are hardcore predicates for one-way functions f.

New Proofs of Old Results

It is easy to see that the hardcore predicates of [5, 18, 59] for candidate one-way functions EXP, RSA, $Rabin$ and ECL, are special cases of the segment predicate defined above. Thus, we re-prove in an entirely different and uniform manner, all the results of [5, 18, 59]

In contrast to previous proofs, the technical essence of the new proofs is to define appropriate codes and to list decode them. These two tasks are independent of the one-way function in question and depend only on the predicate. The only consideration given to the one-way function is for devising a way to access the corrupted codeword (step 3 in the methodology). A task which turns out to be very simple in all the cases considered. We stress that the proofs obtained here are completely different than the previous ones. In particular, the proofs do not require to use the binary gcd algorithm used in previous proofs of the hardcore predicates for RSA [5, 14, 51], nor the square root extraction over finite fields as in [18].

We present a new proof method for simultaneous security of many bits. For this purpose we generalize the notion of balanced hardcore predicates to unbalanced ones. To prove simultaneous bit security, we will show that any violation of simultaneous bit security implies a predictor for some unbalanced hardcore predicate. Using this method we show that a class of functions called *segment functions* – an extension of segment predicates, are simultaneously secure for $RSA, Rabin, EXP$, and ECL. In particular, this implies simultaneous security of the $O(\log\log N)$ most significant bits for those candidate OWFs [64, 86] .

Finally, the new framework applies to proving Goldreich-Levin hardcore predicate in a natural manner where indeed the appropriate code is the Hadamard code.

For a partial summary of these results, see table 1.

Predicate (or function) P	Function f	Code $\mathcal{C}^P = \{C_x\}_x$
$GL(x, r)$	$f(x)\colon \{0,1\}^{\lvert r\rvert} \to \{0,1\}^*$	$\{C_x(i) = GL(x, i)\}_{x\in\{0,1\}^n}$
$msb_N(x)$	$RSA_{N,e}(x) = x^e \bmod N$	$\{C_x(i) = msb_N(x \cdot i \bmod N)\}_{x\in\mathbb{Z}_N^*}$
$msb_{p-1}(x)$	$EXP_{p,g}(x) = g^x \bmod p$	$\{C_x(i) = msb_{p-1}(x \cdot i \bmod p-1)\}_{x\in Z_{p-1}^*}$
$msb_q(x)$	$ECL_{p,Q}(x) = xQ$	$\{C_x(i) = msb_q(x \cdot i \bmod q)\}_{x\in Z_q^*}$
$TriLsb_{p-1}$	$EXP_{p,g}(x) = g^x \bmod p$	$\{C_x(i) = TriLsb_{p-1}(x \cdot i \bmod p-1)\}_{x\in Z_{p-1}^*}$

Table 7.1: Example of predicates (or a function) and codes. Notations details: $GL(z, r) = (-1)^{\langle z,r\rangle}$; $msb_d(z) = 1$ if $0 \le z < \frac{d}{2}$, -1 o/w; q denotes the order of Q; Assuming $p-1$ is co-prime to 3, $TriLsb_{p-1}(x) = msb(x/3)$ (where the division is modulo $p-1$)

On Diffie-Hellman Hardcore Predicates

The fundamental difference between modern cryptography and the classical one is the use of *public key cryptosystems*, namely, cryptosystems in which the communicating parties exchange only *public keys* and need not know each others *private keys*. The first public *key exchange protocol* was suggested by Diffie and Hellman in their seminal paper "New Directions in Cryptography" [25]. In the Diffie-Hellman protocol, the communicating parties, Alice and Bob, each choose private keys $g^a \bmod p$ and $g^b \bmod p$, respectively (for p a prime, and g a generator of Z_p^*), and exchange the public key $g^{ab} \bmod p$.

The Diffie-Hellman key exchange protocol is based on the *Diffie-Hellman (DH) function*

$$DH_{g,p}(g^a, g^b) = g^{ab} \bmod p$$

for p a prime, and g a generator of the group Z_p^*; and its security is based on the *Decisional Diffie-Hellman assumption (DDH)*, which says that no PPT algorithm can distinguish with non-negligible advantage between the two distributions of $\langle g^a, g^b, g^{ab}\rangle$ and $\langle g^a, g^b, g^r\rangle$, where a, b, r are chosen uniformly at random from Z_p^* (and all the exponentiation are modulo p). A –possibly weaker– assumption is the *Computational Diffie-Hellman assumption (CDH)*, which says that there is no probabilistic polynomial time (PPT) algorithm that, given p, g, g^a, g^b returns g^{ab} (where, again, all the exponentiation are modulo p).

The Diffie-Hellman function and assumptions have been widely used as fundamental primitives in many cryptographic applications. Nonetheless, some of the most central and essential relevant questions have remained open (for a survey see [20]). One such fundamental problems is finding a deterministic hardcore predicate for the Diffie-Hellman function.

Goldreich and Levin [38] showed that for any one-way function f, given $f(x)$ and a random string $r \in \{0,1\}^{|x|}$, it is hard to predict $\sum x_i r_i \bmod 2$. Thus, any one-way function, including the Diffie-Hellman function, has a "randomized" hardcore predicate.

What about deterministic hardcore predicates? For example, is it hard to decide the msb_p predicate, namely, whether the secret g^{ab} is greater or smaller than $\frac{p}{2}$?

For conjectured one-way functions, such as $EXP(x) = g^x \bmod p$ or $RSA(x) = x^e \bmod n$, the first deterministic hardcore predicates were discovered roughly twenty years ago [5, 18], and many other deterministic hardcore predicates were discovered since [16,32,41,42,51,59,86,90]. However, *not even one single deterministic hardcore predicates was found for the Diffie-Hellman secret g^{ab}*. A partial result in this direction is the work of Boneh and Venkatesan [19] showing that it is hard to *compute* the $k = O(\sqrt{\log p})$ most significant bits of g^{ab}, given g^a, g^b.

A fundamental question is whether it is possible to improve the result of [19] in: (1) reducing k to one, and (2) showing that even just *predicting* the most significant bit of g^{ab}, given g^a and g^b, with a non-negligible advantage over a random guess cannot be done by a probabilistic polynomial time algorithm, under the CDH assumption. Namely, can we exhibit a deterministic predicate $P\colon \mathbb{Z}_p \to \{\pm 1\}$ such that the existence of a PPT algorithm B that on inputs g^a and g^b returns $P(g^{ab})$ *with non-negligible advantage over a random guess* contradicts the CDH assumption.

We relate the above question to the complexity of the problem of learning characters with noise (LCN), showing that if LCN with random samples access (rLCN) is in BPP, then every segment predicate is hardcore for the Diffie-Hellman function. Furthermore, we show that the latter is true even if easier versions of LCN (such as: LCN with GP-access, that is, with access to samples $\{(x_i, f(x_i))\}_{i=1}^t$ with x_i's a random geometric progression, or LCN with DH-access, that is, with access to samples $f(g^x/g^{ab})$ for any $x = a'b'$ s.t. $g^{a'}, g^{b'}$ can be efficiently computed given g^a, g^b) are in BPP.

These results can be interpreted in two ways. One could either try to find an efficient algorithm for LCN in one of the above access models, with the goal of proving segment predicates are hardcore for the Diffie-Hellman function. Alternatively, one could interpret these results as evidence to the intractability of LCN under those access models.

Other Related Works

Hastad and Naslund [51] showed that the ith bit of x (in its binary representation) is a hardcore predicate for the RSA function. We note that this is different than our result showing that the ith partition bit is a hardcore predicate for the RSA function. It is interesting to study further whether the same techniques can be applied to obtain both sets of results.

Fourier analysis of functions over the Boolean cube $\{0,1\}^n$ has been looked at previously in the context of hardcore predicates in the work of Goldmann *et al* [37].

The literature of hardcore predicates is quite vast and many techniques have been employed throughout the years, which we cannot elaborate on here. The technique of Kaliski [59] for proving hardcore predicates for the discrete logarithm problem in general groups might be another interesting avenue to explore in the context of list decoding for discrete log based functions; it also does not use square root extraction of [18] as well.

Organization of this Chapter

In section 7.2, we formally describe the conditions for proving hardcore predicates via list decoding approach. In section 7.3, we focus on showing hardcore predicates for number theoretic functions. In section 7.4 we use our techniques to prove simultaneous bit security. In section 7.5 we discuss our observations regarding Diffie-Hellman hardcore predicates.

7.2 Hardcore Predicates via List Decoding

In our list decoding approach, to prove that a predicate P is hardcore for a one-way function f, we show that there is a list decodable code such that predicting P with respect to f provides query access to a corrupted codeword encoding x given $f(x)$. In this section we prove that the list decoding approach works, namely, we show that the existence of such a code indeed implies that P is hardcore for f. More precisely, we define "accessible codes", and show that P is hardcore for f, if there is a list decodable code which is accessible with respect to P and f. This is shown in Theorem 7.3.

Throughout this section $\mathcal{F} = \{f_i \colon \mathcal{D}_i \to R_i\}_{i\in I}$ denotes a collection of OWFs, $\mathcal{P} = \{P_i \colon \mathcal{D}_i \to \{\pm 1\}\}_{i\in I}$ denotes a collection of predicates, and $\mathcal{C}^{\mathcal{P}} = \left\{\mathcal{C}^{P_i}\right\}_{i\in I}$ denotes a collection of codes, where $\mathcal{C}^{P_i} = \left\{C_x^{P_i} \colon \mathcal{D}_i \to \{\pm 1\}\right\}$ is a code with codewords $C_x^{P_i}$ corresponding to a non-negligible fraction of the $x \in \mathcal{D}_i$.

We define *accessible codes.* The central property of accessible codes w.r. to $\mathcal{P}$ and $\mathcal{F}$ is that, given $f(x)$ and an algorithm that predicts P with respect to f, query access to a corrupted codeword encoding x can be gained. This is proved in Lemma 7.3.1.

Definition 7.1 (Accessible)**.** *Let $\mathcal{P}$ be a collection of predicates. We say that $\mathcal{C}^{\mathcal{P}}$ is* accessible w.r. to *$\mathcal{F}$, if there exists a PPT access algorithm A, s.t. $\forall i \in I \cap \{0,1\}^k$, $\mathcal{C}^{P_i}$* is accessible w.r. to *f_i, namely,*

1. **Code access***: $\forall x, j \in \mathcal{D}_i$, $A(i, f_i(x), j)$ returns $f_i(x')$ s.t. $C_x^{P_i}(j) = P_i(x')$.*

2. **Well spread***: For uniformly distributed $C_x^{P_i} \in \mathcal{C}^{P_i}$ and $j \in \mathcal{D}_i$, the distribution of x' satisfying $f_i(x') = A(i, f_i(x), j)$ is statistically close*[1] *to the uniform distribution on $\mathcal{D}_i$.*

3. **Bias preserving***: For every codeword $C_x^{P_i} \in \mathcal{C}^{P_i}$, $\left|\Pr_j[C_x^{P_i}(j) = 1] - \Pr_z[P_i(z) = 1]\right| \leq \nu(k)$, where ν is a negligible function.*

An example of an accessible code is the Hadamard code.

[1]Namely, $\frac{1}{2}\sum_{c\in\mathcal{D}_i} |\Pr_{x'}[x' = c] - \Pr_z[z = c]| < \nu(k)$ for a negligible function ν.

Example 7.2 (Binary Hadamard code is accessible)**.** *Let $\mathcal{F}' = \{f'_n\colon \{0,1\}^n \times \{0,1\}^n \to R_n\}$ denote a collection of OWFs, where $f'_n(x,y) = f_n(x).y$ is the concatenation of $f_n(x)$ and y. Let $GL = \{GL_n\colon \{0,1\}^n \times \{0,1\}^n \to \{\pm 1\}\}$ be the collection of predicates: $GL_n(x,r) = (-1)^{\langle x,r\rangle}$. Let $\mathcal{C}^{GL} = \left\{\mathcal{C}^{GL_n}\right\}$ be the collection of* binary Hadamard codes*:*

$$\mathcal{C}^{GL_n} = \{C_{x,y}\colon \{0,1\}^n \to \{\pm 1\},\ C_{x,y}(j) = (-1)^{\langle x,j\rangle}\}$$

Then the algorithm $A(n, f'_n(x,y), (j',j)) = f_n(x).j$ is an access algorithm for $\mathcal{C}^{GL_n}$ w.r to f'_n, and GL_n is well spread, and bias preserving.

Theorem 7.3 (List Decoding Approach)**.** *Assume a collection of codes $\mathcal{C}^{\mathcal{P}} = \left\{\mathcal{C}^{P_i}\right\}_{i\in I}$, s.t. $\mathcal{C}^{\mathcal{P}}$ is list decodable and accessible w.r. to $\mathcal{F}$, then $\mathcal{P}$ is hardcore of $\mathcal{F}$.*

Proof. It suffices to show for a non-negligible fraction of the indexes $i \in I$ a reduction of inverting f_i with non-negligible probability to predicting P_i from f_i. For ease of notation, in the rest of the proof we fix some $i \in I \cap \{0,1\}^k$, and drop the indexes.

Assume an algorithm B that predicts P from f. By lemma 7.3.1, there exists a non-negligible function ρ and a non-negligible fraction of the codewords $C^P_x \in \mathcal{C}^P$ s.t. we have query access to a corrupted codeword w_x satisfying $\Delta(w_x, C^P_x) \leq \mathsf{minor}_{C^P_x} - \rho(k)$.

We list-decode w_x to obtain a short list L containing x. Evaluating f on every candidate x' in L we output x' such that $f(x') = f(x)$ thus inverting $f(x)$.

For ease of notations, we fix some $i \in I \cap \{0,1\}^k$ and drop the indexes.

Lemma 7.3.1. *Let $P\colon \mathcal{D} \to \{\pm 1\}$ be a predicate. Assume $\mathcal{C}^P$ is accessible w.r. to f. Assume we are given a PPT algorithm B that predicts P from f. Then, for a non-negligible fraction of the codewords $C^P_x \in \mathcal{C}^P$, given $f(x)$, we have query access to a corrupted codeword w_x satisfying*

$$\Delta(w_x, C^P_x) \leq \mathsf{minor}_{C^P_x} - \rho(k)$$

for ρ a non-negligible function. In particular, for balanced $\mathcal{P}$, $\Delta(w_x, C^P_x) \leq \frac{1}{2} - \rho(k)$.

Proof. As $\mathcal{C}^P$ is accessible w.r. to f, there exists an access algorithm A as in definition 7.1.

Given $f(x)$, define[2] w_x by

$$w_x(j) = B(A(f(x), j))$$

Let $a_{x,j} \in \mathcal{D}$ satisfy $f(a_{x,j}) = A(f(x), j)$. Since the code is well spread, and B predicts P with some non-negligible advantage ρ',

$$\Pr[B(f(a_{x,j})) = P(a_{x,j})] \geq \mathsf{maj}_P + \rho'(k) - \nu(k)$$

[2]Although A, B are probabilistic algorithms, w.l.o.g we assume w_x is well-defined: Since our list decoding algorithm accesses w_x only polynomially many times, by taking $w_x(j)$ to be the majority value in polynomially many applications of A, B, we have only a negligible probability to of encountering different values for the same entry $w_x(j)$.

where ν is a negligible function, and the probability is taken over random coin tosses of B and over random choices of $C_x^P \in \mathcal{C}^P$ and $j \in \mathcal{D}$.

Let $2\rho(k) = \rho'(k) - \nu(k)$. Thinking of $a_{x,j}$ as first choosing $C_x \in \mathcal{C}^{\mathcal{P}}$, and then $j \in \mathcal{D}$, by a counting argument, $\exists S' \subseteq \mathcal{C}^P$ of at least $\rho(k)$ fraction of the codewords s.t.

$$\forall C_x^P \in S',\ \Pr[B(f(a_{x,j})) = P(a_{x,j})] \geq \mathsf{maj}_P + \rho(k)$$

where the probability is taken over random coin tosses of B and over random choices of $j \in \mathcal{D}$.

Now, as the code is bias preserving $\left|\mathsf{maj}_{C_x} - \mathsf{maj}_P\right| \leq \nu'(k)$, where ν' is a negligible function. Therefore, there exists a non-negligible function $\rho' = \rho - \nu'$ s.t.

$$\forall C_x^P \in S',\ \Pr[B(f(a_{x,j})) = P(a_{x,j})] \geq \mathsf{maj}_{C_x} + \rho'(k)$$

Namely, $\forall C_x^P \in S'$, $\Delta(w_x, C_x^P) \leq \mathsf{minor}_{C_x} - \rho'(k)$. □ □

Remark 7.4. *The above theorem hold even if only for non-negligible fraction of the indexes I, $\mathcal{C}^{P_i}$ list decodable and accessible w.r. to f_i*

Remark 7.5. *Since our list decoding algorithm accesses the corrupted codeword w_x only polynomially many time, it cannot distinguish between two codewords which are within negligible distance from each other. Consequently, our proof of $\mathcal{P}$ being hardcore for $\mathcal{F}$ implies that $\mathcal{P}' = \{P_i'\}_{i \in I}$ is also hardcore for $\mathcal{F}$, as long as each codeword $C_x \in \mathcal{C}^{P_i'}$ is within negligible distance from the codeword $C_x \in \mathcal{C}^{P_i}$.*

Extension to Weak Predictors

Theorem 7.3 exclude the existence of a predictor algorithm to $\mathcal{P}$ w.r. to $\mathcal{F}$. Theorem 7.3 can be generalized to exclude also a weaker "predictor" B that guarantees *only for x s.t. $P(x) = 1$* to predict P better than a random guess. Such weak predictors come up in our analysis of simultaneous bits security.

In the following we formally define such weak predictors, rephrase Theorem 7.3 accordingly and adapt its proof.

Definition 7.6. *We say that B is a* weak predictor *of $\mathcal{P}$ w.r. to $\mathcal{F}$, if for a non-negligible fraction of the $i \in I$,*

$$Pr[B(f_i(x)) = P_i(x) | P_i(x) = 1] > \Pr[B(f_i(x)) = 1] + \rho(k)$$

for ρ a non-negligible function and where the probability is taken over the random coins of the algorithm B and over the random choice of $x \in \mathcal{D}_i \cap \{0,1\}^k$.

Theorem 7.7 (Weak predictor variant of Theorem 7.3)**.** *Let $\mathcal{C}^{\mathcal{P}} = \left\{\mathcal{C}^{P_i}\right\}_{i \in I}$ be a collection of codes that are accessible w.r. to $\mathcal{F}$ and list decodable. Then there exists no weak predictor of $\mathcal{P}$ w.r. to $\mathcal{F}$.*

Proof. Fix $i \in I$ s.t. $\mathcal{C}^{P_i}$ is accessible w.r. to f_i and list decodable. We show an algorithm that for a non negligible fraction of the $x \in \mathcal{D}_i$, given $f(x)$ and query access to B, returns x.

In the Lemma below we show that given $f(x)$ and query access to B, we can gain query access to a corrupted codeword w_x s.t. $\langle w_x, C_x\rangle \geq \widehat{w_x}(0) + \rho(k)$.

This implies that $\langle w_x, C_x\rangle \geq \widehat{w_x}(0)\widehat{C_x}(0) + \rho$, because $\widehat{C_x}(0) \leq 1$ for binary codes. Now since $\mathcal{C}^{\mathcal{P}}$ is list decodable, then given such w_x, there is an efficient algorithm that finds a short list L containing x.

Evaluating f on every candidate $x' \in L$ we then output x' s.t. $f(x') = f(x)$.

Lemma 7.7.1 (Weak predictor variant of Lemma 7.3.1)**.** *Let $P\colon \mathcal{D} \to \{0,1\}$ be a predicate. Assume $\mathcal{C}^P$ is accessible w.r. to $\mathcal{F}$. Assume we are given a PPT algorithm B s.t. B is a weak predictor for $\mathcal{P}$ w.r. to $\mathcal{F}$ Then, for a non-negligible fraction of the codewords $C_x \in \mathcal{C}^{P_N}$, given $f_N(x)$, we can gain query access to a corrupted codeword w_x satisfying*

$$\langle w_x, C_x\rangle \geq \widehat{w_x}(0) + \rho(k)$$

for ρ a non-negligible function and $k = \log N$.

Proof. Recall that for a weak predictor B, $Pr[B(f_i(x)) = P_i(x) | P_i(x) = 1] > \Pr[B(f_i(x)) = 1] + \rho(k)$. We change notation so that P_i, B accept values in $0,1$ (where 1 stays 1, and each value different than 1 is changed to 0). In these notations, a weak predictor B satisfies that for a non negligible fraction of the $i \in I$,

$$\mathbb{E}[B(f_i(x)) \cdot P_i(x)] > \widehat{B}(0) + \rho(k) \qquad (7.1)$$

for ρ a non negligible function and where the probability is over the random coins of B and over the choice of $x \in \mathcal{D}_i \cap \{0,1\}^k$. In the following we fix some i and drop indexes.

The proof now follows similarly to the proof of Lemma 7.3.1 above. Let us elaborate. Define

$$w_x(j) = B(A(f(x), j))$$

Let $a_{x,j}$ satisfy $f(a_{x,j}) = A(f(x), j)$. Because the code is well spread then $|\mathbb{E}_{x,j\in\mathcal{D}}[B(A(f(x),j)) \cdot P(a_{x,j})] - \mathbb{E}_{x\in\mathcal{D}}[B(f(x)) \cdot P(x)]| < \nu$ and $\left|\widehat{B}(0) - \mathbb{E}_{x\in\mathcal{D}}[\widehat{w_x}(0)]\right| < \nu$ for a negligible function ν. By Equation 7.1, this implies that

$$\mathop{\mathbb{E}}_{x,j\in\mathcal{D}}[B(A(f(x),j)) \cdot P_N(a_{x,j})] > \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\widehat{w_x}(0)] + \rho'(k)$$

for ρ' a non-negligible function. Namely,

$$\mathop{\mathbb{E}}_{x\in\mathcal{D}}\langle w_x, C_x\rangle \geq \mathop{\mathbb{E}}_{x\in\mathcal{D}} \widehat{w_x}(0) + \rho'$$

(where we rewrote $\mathbb{E}_{x,j\in\mathcal{D}}[B(A(f(x),j)) \cdot P(a_{x,j})]$ as $\mathbb{E}_{x\in\mathcal{D}}\,\mathbb{E}_{j\in\mathcal{D}}[B(A(f(x),j)) \cdot P(a_{x,j})] = \mathbb{E}_{x\in\mathcal{D}}\langle w_x, C_x\rangle$).

Finally, by a counting argument for a non negligible fraction of the $x \in \mathcal{D}$,

$$\langle w_x, C_x\rangle \geq \widehat{w_x}(0) + \rho'(k)/2$$

□ □

7.3 Number Theoretic Hardcore Predicates

Let $\mathbb{Z}_N$ be the ring of integers with addition and multiplication modulo N. In this section we prove that a broad family of predicates over $\mathbb{Z}_N$, named: *segment predicates*, are hardcore for the candidate one-way functions EXP, RSA, *Rabin* and ECL. The definition of segment predicates includes as a special case predicates previously shown hardcore [5, 18] as well as other predicates not previously known to be hardcore.

7.3.1 Segment Predicates

A segment predicate is any arbitrary assignment of Boolean values to an arbitrary partition of $\mathbb{Z}_N$ into $poly(\log N)$ segments, or a multiplicative shift of such an assignment. A segment predicate can be balanced (with the same number of 0's and 1's) or unbalanced as long as it is far from a constant function.

Definition 7.8 (Segment Predicate). *Let $\mathcal{P} = \{P_N : \mathbb{Z}_N \to \{\pm 1\}\}$ be a collection of predicates that are non-negligibly far from constant, namely, $\exists$ non-negligible function ρ s.t. $\mathsf{maj}_{P_N} \leq 1 - \rho(k)$ where $k = \log N$.*

- *We say that P_N is a* basic t-segment predicate *if $P_N(x+1) \neq P_N(x)$ for at most t x's in $\mathbb{Z}_N$.*
- *We say that P_N is a* t-segment predicate *if there exist a basic t-segment predicate P' and $a \in \mathbb{Z}_N$ which is co-prime to N s.t. $\forall x \in \mathbb{Z}_N, P_N(x) = P'(x/a)$.*
- *If $\forall N$, P_N is a $t(N)$-segment predicate, where $t(N)$ is polynomial in* $\log N$*, we say that $\mathcal{P}$ is a collection of* segment predicates.

Remark 7.9. *Requiring that $\mathcal{P}$ is non-negligibly far from constant is not essential. If it is close to constant then, trivially, $P_i(x)$ cannot be predicted with a* **non-negligible advantage** *over guessing its majority value, as the majority guess is already extremely good.*

The definition of segment predicates is quite general. It captures many of the previous predicates considered for RSA, *Rabin*, EXP and ECL as well as new predicates. In the following we illustrate the generality and ease of working with the segment predicate definition.

Examples of Segment Predicates

Most-significant bit is a segment predicate. Let $msb : \mathbb{Z}_N \to \{\pm 1\}$ be defined by $msb(x) = 1$ if $x < N/2$, and -1 otherwise. This is a basic 2-segment predicate, since it changes value only twice.

Least-significant of RSA is a segment predicate. Let $lsb : \{0, ..., N-1\} \to \{\pm 1\}$ be defined by $lsb(x) = 1$ iff x is even. When N is odd, $\forall x$, $lsb(x) = msb(x/2)$, thus as msb is a basic 2-segment predicate, lsb is a 2-segment predicate with $a = 2$. Consequently, lsb is a segment predicate for RSA, as well as for any other function over $\mathbb{Z}_N$, where N is odd; but it is not a segment predicate for functions over an even domain such as EXP (where the domain Z_{p-1} is even, since p is prime).

Partition bits of RSA are segment predicates. We define the i-th partition bit, $b_i\colon \mathbb{Z}_N \to \{\pm 1\}$, to give alternating values on a partition of $\mathbb{Z}_N$ to 2^i intervals, namely, $b_i(x) = msb(x2^i)$. This is a natural generalization of the most-significant bit (corresponding to $i = 0$), in which $\mathbb{Z}_N$ is partitioned into two halves. Again as $msb(x)$ is a basic 2-segment predicate, b_i is a 2-segment predicate with $a = 2^{-i}$ for RSA, as well as for any other function over $\mathbb{Z}_N$, where N is odd.

Simultaneous bits. Consider an algorithm distinguishing a specific assignment for the $O(\log\log N)$ most significant bits from a random assignment. This is a segment predicate, since each assignment for those bits defines an interval I in $\mathbb{Z}_N$, and a distinguisher for those bits can be thought of as a basic 2-segment predicate P such that $P(x) = -1$ iff $x \in I$. Similarly, other simultaneous bits correspond to general segment predicates. Let us note that when the number of bits is greater than $O(\log\log N)$, then P is within negligible distance from a constant function, and thus it is not a segment predicate.

Example of a new basic segment predicate. In general, we can define a new basic segment predicate by choosing any partition of $\mathbb{Z}_N$ to polynomially many intervals, and giving arbitrary answer (in ± 1) for each interval. For example, a segment predicate might give 1 on the first 10% of the inputs and between the middle $50-90\%$, and -1 otherwise.

Example of a new segment predicate. From any basic segment predicate, and any a co-prime to N one may define a (general) segment predicate. For example, in the EXP case, though the lsb is not a segment predicate, many others are. For instance, consider a trinary partition of the msb defined by $TriLsb(x) \overset{def}{=} msb(x/3)$. Under the standard assumption that $p - 1 = 2q$, where q is prime, this is a segment predicate.

7.3.2 Proving Segment Predicates Hardcore

We prove that any segment predicate is hardcore for any one-way function f defined over $\mathbb{Z}_N$ for which, for a non-negligible fraction of the x's, given $f(x)$ and y, one can efficiently compute $f(xy)$ (where xy is multiplication in $\mathbb{Z}_N$). This includes the functions $EXP_{p,g}$, $RSA(x)$, $Rabin(x)$, and $ECL_{a,b,p,Q}$ (naturally the appropriate N in each case differs).

To prove this theorem, we apply our framework presented in Theorem 7.3. Namely, let $\mathcal{F}$ be a collection of OWFs, we prove $\mathcal{P}$ hardcore of $\mathcal{F}$ as follows:

1. We exhibit a collection of codes $\mathcal{C}^{\mathcal{P}} = \left\{\mathcal{C}^{P_N}\right\}$, each code $\mathcal{C}^{P_N}$ consisting of codewords $C_x\colon \mathbb{Z}_N \to \{\pm 1\}$ for a non-negligible fraction of $x \in \mathbb{Z}_N$,

2. We show that $\mathcal{C}^{\mathcal{P}}$ is list decodable, and

3. We show that $\mathcal{C}^{\mathcal{P}}$ is accessible with respect to $\mathcal{F}$

Interestingly, all the above tasks —except showing $\mathcal{C}^{\mathcal{P}}$ is accessible— are independent of the functions $\mathcal{F}$. Looking ahead, we note that showing $\mathcal{C}^{\mathcal{P}}$ is accessible w.r. to the (conjectured) OWFs *RSA*, *Rabin*, *EXP* and *ECL*, is very simple.

Theorem 7.10. *Let $\mathcal{P} = \{P_N \colon \mathbb{Z}_N \to \{\pm 1\}\}$ be a collection of segment predicates. Then, $\mathcal{P}$ is hardcore for RSA, Rabin, EXP, ECL, under the assumption that these are OWFs.*

Proof. To prove the theorem we first define the code we use in Definition 7.11, then show it is list decodable in Lemma 7.12, and prove it is accessible with respect to *RSA*, *Rabin*, *EXP* and *ECL* in lemmata 7.14-7.22. Combined with Theorem 7.3 this concludes the proof of the theorem. □

Defining a Code

The code we use is the Multiplication code studied in Chapter 6. We repeat here its definition in the context of number theoretic hardcore predicates.

Definition 7.11 (Multiplication code $\mathcal{C}^{\mathcal{P}}$)**.** *For each predicate $P_N \colon \mathbb{Z}_N \to \{\pm 1\}$, we define the* multiplication code $\mathcal{C}^{P_N} = \{C_x \colon \mathbb{Z}_N \to \{\pm 1\}\}_{x \in \mathbb{Z}_N^*}$ *by*

$$C_x(j) = P_N(j \cdot x \bmod N)$$

For a collection of predicates $\mathcal{P} = \{P_N \colon \mathbb{Z}_N \to \{\pm 1\}\}$, denote $\mathcal{C}^{\mathcal{P}} = \left\{\mathcal{C}^{P_N}\right\}$.

Note that $\mathcal{C}^{P_N}$ consists of codewords C_x for a non-negligible fraction of the $x \in \mathbb{Z}_N$ (since $\mathbb{Z}_N^*$ is a non-negligible fraction of $\mathbb{Z}_N$).

List Decoding

We show that the Multiplication code $\mathcal{C}^{\mathcal{P}}$ is list decodable.

Lemma 7.12 (List decoding)**.** *Let $\mathcal{P}$ be a segment predicate and $\mathcal{C}^{\mathcal{P}}$ the Multiplication Code for $\mathcal{P}$, then $\mathcal{C}^{\mathcal{P}}$ is list decodable.*

Proof. In Chapter 6, we showed that the Multiplication codes $\mathcal{C}^{\mathcal{P}}$ are list decodable if $\mathcal{P}$ is well concentrated. In Claim 7.12.1 show that segment predicates $\mathcal{P}$ are well concentrated. Thus we conclude that $\mathcal{C}^{\mathcal{P}}$ is list decodable for every segment predicate $\mathcal{P}$.

Recall that $\mathcal{P}$ is *well concentrated* if $\forall N \in \mathbb{N}, \varepsilon > 0$, $\exists \Gamma \subseteq \mathbb{Z}_N$ s.t. (i) $|\Gamma| \leq \mathit{poly}(\log N/\varepsilon)$, (ii) $\|f_N - f_{N|\Gamma}\|_2^2 \leq \varepsilon$, and (iii) for all $\alpha \in \Gamma$, $\mathit{gcd}(\alpha, N) \leq \mathit{poly}(\log N/\varepsilon)$ (where $\mathit{gcd}(\alpha, N)$ is the greatest common divisor of α and N).

Claim 7.12.1. *Let $\varepsilon > 0$. For a t-segment predicate $P \colon Z_N \to \{\pm 1\}$, P is concentrated within ε on $\Gamma = \{\chi_\alpha | \mathsf{abs}(\alpha) \leq O(t^2/\varepsilon)\}$, i.e*

$$\|P_{|\{\chi_\alpha \,|\mathsf{abs}(\alpha) > O(t^2/\varepsilon)\}}\|_2^2 \leq \varepsilon$$

Proof. Let us first examine the Fourier coefficients of a basic 2-segment predicate P, which gives 1 on a segment I and -1 otherwise:

$$\left|\widehat{P}(\alpha)\right| = \mathop{\mathbb{E}}_{x}[P(x)\chi_\alpha(x)] = \frac{1}{N}[\sum_{x\in I}\chi_\alpha(x) - \sum_{x\notin I}\chi_\alpha(x)]$$

By Proposition 3.33, $\left|\frac{1}{N}\sum_{y=0}^{l-1}\chi_\alpha(y)\right| < \frac{1}{\mathsf{abs}(\alpha)}$. Consequently, since $\widehat{P}(\alpha)$ is the difference of two sums $\sum\chi_\alpha(x)$, and each can be expressed as a difference of two such sums initiated at 0, then $\left|\widehat{P}(\alpha)\right| < O(1/\mathsf{abs}(\alpha))$.

Now, let us examine a basic t-segment predicate P. A basic t-segment predicate defines a partition of Z_N into t segment I_j, so that P is constant on each segment I_j. Thus we can express P as a sum, $P = t-1+\sum_{j=1}^{t}P_j$, of functions $P_j\colon Z_N \to \{\pm 1\}$ such that $P_j(x)$ is the constant $P(x)$ for $x\in I_j$ and -1 otherwise.

Note that each P_j is a basic 2-segment predicate, thus $\left|\widehat{P_j}(\alpha)\right| < O(1/\mathsf{abs}(\alpha))$. Therefore,

$$\left|\widehat{P}(\alpha)\right| = \left|\sum_{j=1}^{t}\widehat{P_j}(\alpha)\right| \le O(t/\mathsf{abs}(\alpha))$$

Now, consider the sum of weights over all large characters χ_α:

$$\sum_{\mathsf{abs}(\alpha)>k}\left|\widehat{P}(\alpha)\right|^2 \le O(t^2)\sum_{\mathsf{abs}(\alpha)>k}\frac{1}{\mathsf{abs}(\alpha)^2} < O(t^2/k)$$

which implies that for $\varepsilon > 0$

$$\|P_{|\{\chi_\alpha\ |\mathsf{abs}(\alpha)>O(t^2/\varepsilon)\}}\|_2^2 \le \varepsilon$$

Finally, for a general segment predicate $P_N(x) = P_N{}'(x/a)$ for $P_N{}'$ a basic t-segment predicate and a co-prime to N. By Theorem 2.2, this implies that $\forall\alpha$, $\widehat{P_N}(\alpha) = \widehat{P'_N}(\alpha b)$. So the above implies that P_N is concentrated to within ε on

$$\Gamma = \left\{\chi_\beta \mid \beta = \alpha(x/a) \bmod N, \mathsf{abs}(\alpha) \le O(t^2/\varepsilon)\right\}$$

Since a is co-prime to N then $\max_{\beta\in\Gamma'} gcd(\beta,N) = \max_{\mathsf{abs}(\alpha)\le O(t^2/\varepsilon)} gcd(\alpha,N) \le O(t^2/\varepsilon)$. Put together, P_N is well concentrated. □

□

Accessibility

We now show that that the multiplication code $\mathcal{C}^{P_N} = \{C_x\}_{x\in Z_N^*}$ is accessible w.r. to the candidate one-way functions of RSA, $Rabin$, EXP and ECL. As it turns out this task is simple for all these functions.

Accessibility w.r. to RSA

We formally define the RSA collection of functions, and show that the code $\mathcal{C}^{\mathcal{P}}$ is accessible w.r. to RSA.

Definition 7.13 (RSA)**.** *Assuming hardness of inverting the RSA function yields the following collection of OWFs. Define* $RSA = \{RSA_{n,e}(x) = x^e \ mod\ n,\ RSA_{n,e} \colon Z_n^* \to Z_n^*\}_{\langle n,e\rangle \in I}$ *for* $I = \{\langle n, e\rangle,\ n = pq,\ |p| = |q|,\ p \text{ and } q \text{ are primes and } (e, \phi(n)) = 1\}$

One technical issue we need to address, when considering segment predicates in the context of RSA or $Rabin$, is that segment predicates were defined over the domain Z_N whereas $RSA, Rabin$ are defined over Z_N^*. To overcome this difficulty we extend the definition of segment predicates, by saying that $\mathcal{P}' = \{P'_N\} : Z_N^* \to \{\pm 1\}$ is a collection of segment predicates, if there exists a collection $\mathcal{P} = \{P_N \colon Z_N \to \{\pm 1\}\}$ of segment predicate, such that P'_N is the restriction of P_N to inputs from the domain Z_N^*. W.l.o.g assume $P_N(x) = 0$ for every $x \in Z_N \setminus Z_N^*$.

Lemma 7.14 (Accessibility w.r. to RSA)**.** *Let* $\mathcal{P} = \left\{ P_{N|Z_N^*} \,\middle|\, P_N \colon Z_N \to \{\pm 1\} \right\}$ *be a collection of segment predicates, then* $\mathcal{C}^{\mathcal{P}}$ *is accessible w.r. to RSA.*

Proof. Let the access algorithm A be

> **input:** $\langle N, e\rangle, RSA_{N,e}(x), j$
> **output:** If $j \in Z_N^*$, return $RSA_{N,e}(jx) = RSA_{N,e}(j) \cdot RSA_{N,e}(x) \bmod N$;
> else return 0.

For any fixed $x \in Z_N^*$, and uniformly distributed $j \in Z_N$, consider the distribution of x' satisfying $RSA_{N,e}(x') = A(i, RSA_{N,e}(x), j)$. This distribution is close to uniform, because its restriction of this distribution to Z_N^* is uniform, and w.l.o.g there is only a negligible fraction of inputs $j \notin Z_N^*$ (otherwise, RSA can be broken). Therefore, the code is well spread, and bias preserving. □

Accessibility w.r. to $Rabin$

We formally define the $Rabin$ collection of functions, and show that the code $\mathcal{C}^{\mathcal{P}}$ is accessible w.r. to $Rabin$.

Definition 7.15 ($Rabin$)**.** *Assuming hardness of factoring yields the following collection of OWFs. Define* $Rabin = \{Rabin_n(x) = x^2 \ mod\ n,\ Rabin_n \colon Z_n^* \to Z_n^*\}_{n \in I}$ *for* $I = \{n,\ n = pq,\ |p| = |q|,\ p \text{ and } q \text{ are primes and } \}$

Lemma 7.16 (Accessibility w.r. to $Rabin$)**.** *Let* $\mathcal{P} = \left\{ P_{N|Z_N^*} \,\middle|\, P_N \colon Z_N \to \{\pm 1\} \right\}$ *be a collection of segment predicates, then* $\mathcal{C}^{\mathcal{P}}$ *is accessible w.r. to Rabin.*

Proof. Same proof as for RSA. □

Accessibility w.r. to EXP

We formally define the exponentiation functions, denoted EXP, (that is, the inverse of the Discrete Logarithm function), and show that the code $\mathcal{C}^{\mathcal{P}}$ is accessible w.r. to EXP.

Definition 7.17 (EXP)**.** *Assuming hardness of solving the Discrete Log Problem yields the following collection of OWFs. Define* $EXP = \{EXP_{p,g}(x) = g^x \bmod p,$ $EXPp, g\colon$
$Z_{p-1} \to Z_p^*\}_{\langle p,g\rangle \in I}$ *for* $I = \{\langle p, g\rangle,$ *p prime, g generator for* $Z_p^*\}$.

Lemma 7.18 (Accessibility w.r. to EXP)**.** *Let* $\mathcal{P} = \{P_{p,g}\colon Z_{p-1} \to \{\pm 1\}\}$ *be a collection of segment predicates, then* $\mathcal{C}^{\mathcal{P}}$ *is accessible w.r. to* EXP.

Proof. Let the access algorithm A be

> **input:** $\langle p, g\rangle, EXP_{p,g}(x), j$
> **output:** Return $EXP_{p,g}(xj) = EXP_{p,g}(x)^j \bmod N$.

For any fixed $x \in Z_{p-1}^*$, and uniformly distributed $j \in Z_{p-1}$, the distribution of x' satisfying $EXP_{p,g}(x') = A(i, EXP_{p,g}(x), j)$ is uniform. Therefore, the code is well spread, and bias preserving. □

Remark 7.19. *So far, we invert* $EXP_{p,g}(x)$ *only for x's which are co-prime to* $p-1$ *(as* $\mathcal{C}^P = \{C_x\}_{x \in Z_{p-1}^*}$*). Nonetheless, by random self reducibility of EXP we can invert for any x: For* $r \in_R Z_{p-1}$, $EXP_{p,g}(x+r) = g^x \cdot g^r$ *is a generator of* Z_p^* *w.h.p. In this case, we can invert* $EXP_{p,g}(x+r)$ *to find* $x' = x + r$*, and return* $x = x' - r$.

Remark 7.20. *Interestingly, lsb is a segment predicate over odd domains, and thus hardcore for RSA, while it is easy for EXP (where the domain is even). To see where the proof fails, consider the code* $\mathcal{C}^{lsb}$ *consisting of codewords* $C_x(j) = lsb(jx)$ *over an even domain.*

This code has no recovery algorithm with a succinct output, as its codewords correspond only to one out of two functions: $C_x(j) = lsb(jx) = 1$ *for even x's, and* $C_x(j) = lsb(jx) = lsb(j)$ *for odd x's. Moreover, no alternative list decoding algorithm exists, since each codeword is at distance* 0 *from half the codewords.*

Accessibility w.r. to ECL

We formally define the elliptic curve discrete logarithm functions, denoted ECL, and show that the code $\mathcal{C}^{\mathcal{P}}$ is accessible w.r. to ECL.

Definition 7.21 (ECL)**.** *Let* $E_{a,b}(Z_p)$ *denote the additive group of* $n = 2q$ *points on an elliptic curve* $y^2 = x^3 + ax + b$ *over* Z_p*, where* $a, b \in Z_p$, $4a^3 + 27b^2 \neq 0$ *and p is prime.*

Define $ECL = \{ECL_{p,a,b,Q}(x) = xQ,\ ECL_{p,a,b,Q}\colon Z_q \to E_{a,b}(Z_p)\}_{\langle p,a,b,Q\rangle \in I}$ *for* $I = \{\langle p, a, b, Q\rangle$*, where* $Q \in E_{a,b}(Z_p)$ *is a point of order q, and p, a, b satisfy the above conditions*$\}$.

Lemma 7.22 (Accessibility w.r. to ECL)**.** *Let $\mathcal{P} = \{P_{p,a,b,Q} \colon Z_q \to \{\pm 1\}\}$ (where q is the order of the point Q on the elliptic curve $E_{a,b}(Z_p)$) be a collection of segment predicates, then $\mathcal{C}^{\mathcal{P}}$ is accessible w.r. to ECL.*

Proof. Let the access algorithm A be

```
    input: $ECL_{p,a,b,Q}(x), j$
    output: Return $ECL_{p,a,b,Q}(jx) = jECL_{p,Q}(x)$.
```

For any fixed $x \in Z_q$, and uniformly distributed $j \in Z_q$, the distribution of x' satisfying $ECL_{p,a,b,Q}(x') = A(i, ECL_{p,a,b,Q}(x), j)$ is uniform. Therefore, the code is well spread, and bias preserving. □

7.4 Simultaneous Bits Security

We present a new method for proving simultaneous security of many bits: We generalize the notion of balanced hardcore predicates to unbalanced ones, and show that any violation of simultaneous bit security leads to a contradiction by implying a predictor for some unbalanced hardcore predicate. Using this method we show that a class of functions called *segment functions* are simultaneously secure for $RSA, Rabin, EXP$, and ECL. In particular, this implies simultaneous security of the $O(\log\log N)$ most significant bits for those candidate OWFs [64, 86] .

In the following we first formally define simultaneous security of bits using the notion of *hardcore functions*. We then extend the definition of segment predicates to the notion of *segment functions*, and present examples of segment functions. Finally, we show that segments functions are hardcore for $RSA, Rabin, EXP$, and ECL.

Simultaneous security of bits can be captured by the notion of hardcore functions. We say that h is a hardcore function for f if $h(x)$ is hard to predict from $f(x)$. This extends the notion of hardcore predicates from Boolean predicates P to functions h with larger range.

Definition 7.23 (Hardcore function)**.** *Let $\mathcal{F} = \{f_i \colon \mathcal{D}_i \to R_i\}_{i\in I}$ be collections of OWFs. Let $\mathcal{H} = \left\{h_i \colon \mathcal{D}_i \to \{0,1\}^{l(i)}\right\}_{i\in I}$ be collections of functions s.t. for each $i \in I \cap \{0,1\}^k$, $l(i)$ is polynomial in k. We say that a PPT algorithm D is a* distinguisher *for $\mathcal{H}$ w.r. to $\mathcal{F}$, if $\exists$ a non-negligible function ρ s.t.*

$$|\Pr[D(f_i(x), h_i(x)) = 1] - \Pr[D(f_i(x), h(r)) = 1]| > \rho(k)$$

where the probability is taken over the random coin tosses of D and choices of $i \in I \cap \{0,1\}^k$, $x, r \in \mathcal{D}_i$.

We say that $\mathcal{H}$ is hardcore *for $\mathcal{F}$, if there exists no distinguisher for $\mathcal{H}$ w.r. to $\mathcal{F}$.*

We extend the definition of segment predicates to segment functions.

Definition 7.24 (Segment function)**.** *Let $\mathcal{H} = \left\{h_N \colon Z_N \to \{0,1\}^{l(N)}\right\}_{N\in I}$ be a collection of functions. For each $s \in \{0,1\}^{l(N)}$, define a predicate $P_N^{\mathcal{H},s} \colon Z_N \to \{0,1\}$,*

$P_N^{\mathcal{H},s}(x) = 1$ *if* $h_N(x) = s$ *and* 0 *otherwise. We say that* $\mathcal{H}$ *is a collection of* segment functions*, if* $\mathcal{P} = \left\{P_N^{\mathcal{H},s}\right\}_{N\in I, s\in\{0,1\}^{l(N)}}$ *is a collection of segment predicates.*

We give two examples of segment functions.

Example 7.25 (Most significant bits)**.** *Let* $Pref_N(x)$ *be the* $l(N)$ *most significant bits in a binary representation of* x*.* $\forall s \in \{0,1\}^{l(N)}$*, define the unbalanced predicate* $P_N^s(x) = 1$ *if* $Pref_N(x) = s$*, and* 0 *otherwise. When* $l(N) \le O(\log\log N)$*,* P_N^s *is is a segment predicate (as it is non-negligibly far from constant), thus,* $\mathcal{H} = \left\{Pref_N \colon Z_N \to \{0,1\}^{l(N)}\right\}_N$ *is a collection of segment functions.*

Example 7.26 (Dissection bits)**.** *Let* $a \in Z_N^*$*, and let* $Dissect_{a,N}(x) = Pref_N(x/a)$*. Then when* $l(N) \le O(\log\log N)$*,* $\mathcal{H} = \left\{Dissect_{a,N} \colon Z_N \to \{0,1\}^{l(N)}\right\}_N$ *is a collection of segment functions.*

We show that segment functions are hardcore for for RSA, $Rabin$, EXP and ECL.

Theorem 7.27. *Let* $\mathcal{H} = \left\{h_N \colon Z_N \to \{0,1\}^{l(N)}\right\}_N$ *be a collection of segment functions. Then* $\mathcal{H}$ *is hardcore for RSA, Rabin, EXP and ECL, under the assumption that these are OWFs.*

Proof. Let $\mathcal{F} = \{f_N\}_{N\in I}$ denote one of the candidate OWFs: RSA, $Rabin$, EXP or ECL. Let $\mathcal{P} = \left\{P_N^{\mathcal{H},s}\right\}$ denote the collection of segment predicates that corresponds to $\mathcal{H}$.

Assume for contradiction there exists a distinguisher algorithm for $\mathcal{H}$ with respect to $\mathcal{F}$. In Lemma 7.27.1 we show this implies that there exists a weak predictor B for $\mathcal{P}$ w.r. to $\mathcal{F}$.

But, by Theorem 7.7 there exists no weak predictor for $\mathcal{P}$ w.r. to $\mathcal{F}$ (under the assumption that $\mathcal{F}$ is a collection of OWFs), because the code $\mathcal{C}^{\mathcal{P}}$ is list decodable and accessible w.r. to $\mathcal{F}$ (as we proved in the analysis of number theoretic hardcore predicates).

Thus, for $\mathcal{F}$ a OWF there is no distinguishing algorithm for $\mathcal{H}$ with respect to $\mathcal{F}$. Namely, $\mathcal{H}$ is hardcore for $\mathcal{F}$.

Lemma 7.27.1. *If there is a distinguisher* D *for* $\mathcal{H}$ *with respect to* $\mathcal{F}$*, then there is a weak predictor algorithm* B *for* $\mathcal{P}^{\mathcal{H},s}$ *w.r. to* $\mathcal{F}$*.*

Proof. By definition of a distinguisher for $\mathcal{H}$ w.r. to $\mathcal{F}$,

$$|\Pr[D(f_N(x), h_N(x)) = 1] - \Pr[D(f_N(x), h_N(r)) = 1]|$$

is non-negligible (where the probability is taken over the random coin tosses of D and choices of $N \in I \cap \{0,1\}^k$, $x, r \in \mathcal{D}_N$). This implies that for a non-negligible fraction of the indexes N,

$$|\Pr[D(f_N(x), h_N(x)) = 1] - \Pr[D(f_N(x), r) = 1]| \ge \rho(k)$$

for ρ an non-negligible function and where the probability is taken over random choices of $x \in Z_N$ and $r \in \{0,1\}^{l(N)}$. In the following we fix such an index N, and drop the indexes.

Choose a random $s \in \{0,1\}^{l(N)}$, and define an algorithm B by

$$\forall x,\ B(f(x)) = D(f(x), s)$$

By a counting argument, for at least $\rho/2$-fraction of the $s \in \{0,1\}^{l(N)}$,

$$\rho/2 < \left|\Pr_x[D(f(x), s) = 1|h(x) = s] - \Pr_z[D(f(z), s) = 1]\right|$$

In particular, this holds for the random s chosen above with probability at least $\rho/2$. (See details of the counting argument in the claim below). Without loss of generality assume we can take out the absolute value sign, namely, $\rho/2 < \Pr_x[D(f(x), s) = 1|h(x) = s] - \Pr_z[D(f(z), s) = 1]$ (otherwise we can inverse the roles of 0 and 1).

Now, observing that

$$\Pr_x[D(f(x), h(x)) = 1|h(x) = s] = \Pr_x[B(f(x))|P^s(x) = 1]$$

and that

$$\Pr_z[D(f(z), s) = 1] = \Pr_z[B(f(z)) = 1]$$

We conclude that with probability at lest $\rho/2$ over the choice of $s \in \{0,1\}^{l(N)}$,

$$\Pr_x[B(f(x))|P^s(x) = 1] > \Pr_z[B(f(z)) = 1] + \rho/2$$

Claim. (Details of the counting argument.) For at least $\rho/2$-fraction of the $s \in \{0,1\}^{l(N)}$, $\rho/2 < |\Pr_x[D(f(x), s) = 1|h(x) = s] - \Pr_z[D(f(z), s) = 1]|$.
Proof. We rewrite $\Pr[D(f(x), h(x)) = 1] - \Pr[D(f(x), h(r)) = 1]$ as a sum of conditional probabilities: $\frac{1}{|2^{l(N)}|}\sum_{s\in\{0,1\}^{l(N)}} \Pr[D(f(x), h(x)) = 1|h(x) = s] - \Pr[D(f_N(x), s) = 1]$, concluding that

$$\rho < \frac{1}{|2^{l(N)}|}\sum_{s\in\{0,1\}^{l(N)}} |\Pr[D(f(x), h(x)) = 1|h(x) = s] - \Pr[D(f_N(x), s) = 1]|$$

(where we moved the absolute value to inside the summation since by triangle inequality for every reals $a_i \in \mathbb{R}$, $|\sum a_i| \leq \sum |a_i|$).

The claim now follows from a standard counting argument: Denote by $S \subseteq 2^{l(N)}$ the set good s's s.t. $\forall s \in S$, $\rho/2 < |\Pr_x[D(f(x), s) = 1|h(x) = s] - \Pr_z[D(f(z), s) = 1]|$. Then, $\rho < \frac{1}{|2^{l(N)}|}\sum_{s\in\{0,1\}^{l(N)}} |\Pr[D(f(x), h(x)) = 1|h(x) = s] - \Pr[D(f_N(x), s) = 1]| \leq \frac{|S|}{2^{l(N)}} + (1 - \frac{|S|}{2^{l(N)}})\frac{\rho}{2}$ which in turn implies that $|S| \geq \frac{\rho}{2} \cdot 2^{l(N)}$. $\square$ $\square$ $\square$

7.5 On Diffie-Hellman Hardcore Predicates

The above framework for proving hardcore predicates via list decoding applies to many conjectured one-way functions. Still, one cryptographic function of a very wide appeal and usage alludes this treatment. This is the Diffie-Hellman (DH) function.

In this section we relate on the question of whether there exist deterministic hardcore predicates for DH to the complexity of the problem of learning characters with noise (LCN), showing that if LCN with random samples access (rLCN) is in BPP, then every segment predicate is hardcore for DH. Furthermore, we show that the latter is true even if easier versions of LCN (such as: LCN with GP-access, that is, with access to samples $\{(x_i, f(x_i))\}_{i=1}^t$ with x_i's a random geometric progression, or LCN with DH-access, that is, with access to samples $f(g^x/g^{ab})$ for any $x = a'b'$ s.t. $g^{a'}, g^{b'}$ can be efficiently computed given g^a, g^b) are in BPP.

To prove this result we follow our list decoding methodology using the same Multiplication codes $\mathcal{C}^{\mathcal{P}}$ as used in Section 7.3. We have, however, a different access algorithm, because our code access algorithms are function dependent. This access algorithm has a fundamental difference from algorithms devised for the previously considered candidate OWFs. Specifically, this algorithm only provides *random access* (or, more generally, GP-access or DH-access) to corrupted codewords. This is in contrast to the *query access* provided by our access algorithms w.r. to the previously considered candidate OWFs. This difference calls for a new list decoding algorithm for $\mathcal{C}^{\mathcal{P}}$: an algorithm that can decode even when given only random access (or GP-access or DH-access) to the corrupted codeword. We show that if LCN in these access models is in BPP, then such a list decoding algorithm exists.

The Computational Diffie-Hellman (CDH) assumption asserts that computing for random prime p and generator g of $\mathbb{Z}_p^*$, computing $DH_{p,g}(g^a, g^b) = g^{ab}$ is intractable. This gives rise to the following collection of candidate one-way function $invDH_{p,g}$ whose forward direction is the easy to compute function $invDH_{p,g}(g^a, g^b, g^{ab}) = (g^a, g^b)$ whereas their reverse direction requires computing the DH function, because $invDH_{p,g}^{-1}(g^a, g^b) = (g^a, g^b, g^{ab})$

Definition 7.28 (invDH)**.** *Assuming the hardness of computing the Diffie-Hellman function yields the following collection of OWFs. Define*

$$invDH = \{invDH_{p,g}(g^a, g^b, g^{ab}) = (g^a, g^b),\ invDH_{p,g} \colon \mathbb{Z}_p^3 \to \mathbb{Z}_p^2\}_{\langle p,g\rangle \in I}$$

for $I = \{\langle p, g\rangle,\ p$ prime, g a generator of $\mathbb{Z}_p^*\}$.

The access models that arise in our study of DH functions are defined as follows.

Definition 7.29. *1.* **Random samples access.** *We say that an algorithm A_{rand} provides* random samples access *to a function $w \colon \mathbb{Z}_p \to \{\pm 1\}$ if, given p, g, g^a, g^b, A_{rand} outputs a pair $(s, w(s))$ of entry location s and the value of w on this entry, for s distributed uniformly at random in $\mathbb{Z}_p$.*

2. **GP-access.** *We say that an algorithm A_{GP} provides* Geometric Progression access (GP-access) *to a function $w \colon \mathbb{Z}_p \to \{\pm 1\}$ if, given p, g, g^a, g^b and t inte-*

gers $k_1, \dots, k_t$, A_{GP} outputs t pairs $\{(s_j, w(s_j))\}_{j=1}^t$ of entry locations s_j and the values of $w(s_j)$ on those entries, where $s_1 = s^{k_1}, \dots, s_t = s^{k_t}$ for s distributed uniformly at random in $\mathbb{Z}_p$.

3. **DH-access.** *We say that an algorithm A_{DH} provides* Diffie-Hellman access (DH-access) *to a function $w\colon \mathbb{Z}_p \to \{\pm 1\}$ if, given p, g, g^a, g^b and $g^{a'}, g^{b'}$, A_{DH} outputs $w(s)$ for $s = g^{a'b'-ab}$.*

We relate the complexity of LCN in the above access models to the problem of proving hardcore predicates for *invDH*.

Theorem 7.30. *If LCN with random samples access or GP-access or DH-access is in BPP, then every segment predicate is hardcore for invDH.*

Proof. To prove the theorem consider codes $\mathcal{C}^{\mathcal{P}}$ as defined in Definition 7.11. In lemma 7.31 we show that, for any access model M, if LCN in access model M is in BPP, then the code $\mathcal{C}^{\mathcal{P}}$ is list decodable when given a corrupted codeword in access model M. In lemma 7.32 we show that given p, g, g^a, g^b and an algorithm B that predicts $\mathcal{P}$ from *invDH*, we can gain access in access model M being either random samples access, or GP-access, or DH-access to a corrupted codeword w close to the codeword $C_{g^{ab}}$ encoding g^{ab} in $\mathcal{C}^{\mathcal{P}}$.

Combined together this implies that if LCN in access model M is in BPP, then $\mathcal{P}$ is hardcore predicate for *invDH*. This is because, if there is an algorithm that predicts $\mathcal{P}$ from *invDH*, then the following is an algorithm for inverting *invDH*: Given p, g, g^a, g^b, gain query access to a corrupted codeword w close to the codeword encoding g^{ab}, list decode to find the message g^{ab}, and output (g^a, g^b, g^{ab}). □

List Decoding

We explain how to list decode the code $\mathcal{C}^{\mathcal{P}}$ in random samples access or GP-access or DH-access. models, assuming we are given an algorithm solving LCN in these access models.

In chapter 6 we present our list decoding via learning algorithm for the Multiplication Codes. This algorithm had two steps: (1) it applied the SFT Algorithm 3.4 given query access to w to find its significant Fourier coefficients, and (2) it applied Recovery Algorithm 6.50 on the list of significant Fourier coefficients to find all codewords close to w.

In our current settings we do not have query access to w so we cannot apply the SFT algorithm. Nevertheless, we do have random samples access, GP-access or DH-access. So, if there is an efficient algorithm solving LCN in those access models, we can substitute the SFT Algorithm of step (1) with this algorithm to get a list decoding algorithm for $\mathcal{C}^{\mathcal{P}}$ in those access model.

Lemma 7.31 (List decoding)**.** *Let $\mathcal{P}$ be a segment predicate and $\mathcal{C}^{\mathcal{P}}$ the Multiplication code for $\mathcal{P}$. For any access model M, if LCN in access model M is in BPP, then $\mathcal{C}^{\mathcal{P}}$ is list decodable in access model M.*

Proof. In Chapter 6, we showed that any concentrated an recoverable codes over a learnable domain is list decodable. We showed that $\mathcal{C}^{\mathcal{P}}$ is concentrated and recoverable in Lemma 7.12. The assumption that LCN in access model M is in BPP says that the domain of the code is learnable with respect to access model M. Under this assumption we conclude therefore that $\mathcal{C}^{\mathcal{P}}$ is list decodable in access model M. $\square$

Code Access

In the following we show how given g^a, g^b and an algorithm B predicting P, we can gain access to a corrupted codewords w close to the codeword $C_{g^{ab}}$ encoding g^{ab} in the code $\mathcal{C}^{\mathcal{P}}$, where access can be random samples access or GP-access or DH-access.

Lemma 7.32. *Let p be a prime, g a generator of $\mathbb{Z}_p^*$, and $P\colon \mathbb{Z}_p \to \{\pm 1\}$ a predicate. Assume we are given a PPT algorithm B that predicts P from $invDH_{p,g}$. Then, for a non-negligible fraction of the codewords $C_{g^{ab}} \in \mathcal{C}^P$, given g^a, g^b, there is a corrupted codeword $w\colon \mathbb{Z}_p \to \{\pm 1\}$ and access algorithms A_{rand}, A_{GP} and A_{DH} s.t. (i) w is close to the encoding of g^{ab}, that is,*

$$\Delta(w, C_{g^{ab}}) \leq \mathsf{minor}_{C_{g^{ab}}} - \rho$$

for ρ a non-negligible function,[3] *and (ii) the algorithms A_{rand}, A_{GP} and A_{DH} provide random access, GP-access and DH-access, respectively to w; and their running time is polynomial in $\log p, 1/\rho$.*

Proof. Fix g^a, g^b, and define a corrupted codeword $w\colon \mathbb{Z}_p \to \{\pm 1\}$ close to $C_{g^{ab}}$ by

$$w(j) = B(g^a \cdot g^{i/b}, g^b), \text{ for } g^i = j \bmod p$$

We show that for random g^a, g^b, w is indeed close to $C_{g^{ab}}$. This is because $C_{g^{ab}}(j) = P(j \cdot g^{ab})$, and for a predictor algorithm B it holds that $B(g^a \cdot g^{i/b}, g^b) = P(jg^{ab})$ with non negligible advantage over a random guess.

We define three algorithms A_{rand}, A_{GP} and A_{DH} and show that they provide random samples access, GP-access and DH-access respectively to w:

- $A_{rand}(p, g, g^a, g^b) = \left((g^b)^r, B\left(g^a \cdot g^r, g^b\right)\right)$ for $r \in \mathbb{Z}_p$ chosen uniformly at random.
- $A_{GP}(p, g, g^a, g^b, k_1, \ldots, k_t) = (g^{b^{rk_1}}, B(g^a \cdot g^{rk_1}, g^b)), \ldots, (g^{b^{rk_t}}, B(g^a \cdot g^{rk_t}, g^b))$ for $r \in \mathbb{Z}_p$ chosen uniformly at random.
- $A_{DH}(p, g, g^a, g^b, g^{a'}, g^{b'}) = B(g^{a'}, g^{b'})$.

For a non-negligible fraction of the inputs, A_{rand} provides random samples access to w. This is because $B(g^a \cdot g^r, g^b)$ is the value of w on entry $(g^b)^r$, and $(g^b)^r$ is distributed uniformly at random whenever g^b is a generator of $\mathbb{Z}_p$ (an event which happens with a non negligible probability over the choice of $g^b \in \mathbb{Z}_p$).

[3]In particular, for balanced predicates P, $\Delta(w, C_{g^ab}) \leq \frac{1}{2} - \rho(k)$.

For a non-negligible fraction of the inputs, A_{GP} provides GP-access to w. This is because for any $k \in k_1, \dots, k_t$, $B(g^a \cdot g^{rk}, g^b)$ is the value of w on entry X^k for $X = g^{br}$, and g^{br} is distributed uniformly at random whenever g^b is a generator of $\mathbb{Z}_p$ (an event which happens with a non negligible probability over the choice of $g^b \in \mathbb{Z}_p$).

A_{DH} provides DH-access to w. This follows immediately from the definition of the codeword $C_{g^{ab}}$, because $B(g^{a'}, g^{b'})$ is the value of w on entry $g^{a'b'-ab}$. $\square$

Remark 7.33. *In contrast to the above considered access models, it is not clear how to obtain* query access *to a corrupted codeword close to $C_{g^{ab}}$: Given g^a, g^b, a predictor algorithm B and an entry location j, one would like to produce $g^{i/b}$ for $j = g^i$ and access $w(j)$ by computing $B(g^a \cdot g^{i/b}, g^b)$. It's not however clear how to do this, because producing $g^{i/b}$ seems to require computing $g^i = j \bmod p$, i.e., solving the (believe-to-be hard) Discrete-Log problem.*

Bibliography

[1] Dorit Aharonov and Oded Regev. Lattice problems in np intersect conp. *J. ACM*, 52(5):749–765, 2005.

[2] Adi Akavia and Shafi Goldwasser. Manuscript submitted as an NSF grant, Dec. 2004, awarded CCF-0514167.

[3] Adi Akavia, Shafi Goldwasser, and Samuel Safra. Proving Hard-Core Predicates using List Decoding. In *Proc. of 44th IEEE Annual Symposium on Foundations of Computer Science (FOCS'03)*, pages 146–157. IEEE Computer Society, 2003.

[4] Adi Akavia and Ramarathnam Venkatesan. Perturbation Codes. Workshop II: Locally decodable codes, private information retrieval, privacy-preserving data-mining, and public key encryption with special properties, IPAM, UCLA, October 2006.

[5] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *SIAM J. Computing*, 17:194–209, 1988.

[6] N. Alon and Y. Mansour. 3-discrepancy sets and their application for interpolation of sparse polynomials. *IPL: Information Processing Letters*, 54, 1995.

[7] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–, 1992.

[8] Alp Atici and Rocco A. Servedio. Learning unions of $\omega(1)$-dimensional rectangles. In *ALT*, pages 32–47, 2006.

[9] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31. ACM, 1991.

[10] Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In Christian Choffrut and Thomas Lengauer, editors, *STACS*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 1990.

[11] Amos Beimel and Yuval Ishai. Information-theoretic private information retrieval: A unified construction. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *ICALP*, volume 2076 of *Lecture Notes in Computer Science*, pages 912–926. Springer, 2001.

[12] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-François Raymond. Breaking the $o(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In *Proc. 43rd IEEE Annual Symposium on Foundations of Computer Science (FOCS'02)*, pages 261–270, 2002.

[13] M. Bellare, D. Coppersmith, J. Hastad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. In *Proc. of 36th IEEE Annual Symposium on Foundations of Computer Science (FOCS'95)*, page 432, Washington, DC, USA, 1995. IEEE Computer Society.

[14] M. Ben-Or, B. Chor, and A. Shamir. On the cryptographic security of single RSA bits. In *Proc. 15th ACM Annual Symposium on Theory of Computing (STOC'83)*, pages 421–430, 1983.

[15] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.

[16] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[17] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proc. 22nd ACM Annual Symposium on Theory of Computing (STOC'90)*, pages 73–83, New York, NY, USA, 1990. ACM Press.

[18] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[19] D. Boneh and R. Venkatesan. Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes. *Lecture Notes in Computer Science*, 1109:129–142, 1996.

[20] Dan Boneh. The decision Diffie-Hellman problem. *Lecture Notes in Computer Science*, 1423:48–63, 1998.

[21] Nader H. Bshouty, Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning dnf from random walks. *J. Comput. Syst. Sci.*, 71(3):250–265, 2005.

[22] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[23] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *Structural Information and Communication Complexity, 13th International Colloquium, SIROCCO 2006, Chester, UK, July 2-5, 2006, Proceedings*, pages 280–294, 2006.

[24] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In C. Boyd, editor, *Asiacrypt 2001*, volume 2248 of *LNCS*, pages 157–174. Springer, 2001.

[25] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[26] Irit Dinur and Shmuel Safra. The importance of being biased. In *Proc. of 34th ACM Annual Symposium on Theory of Computing (STOC'02)*, pages 33–42. ACM Press, 2002.

[27] David Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 42(4):1289–1306, April, 2005.

[28] P. Elias. List decoding for noisy channels. Technical Report 335, Research Lab. of Electronics, MIT, September 1957.

[29] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings 1994 ACM SIGMOD Conference, Mineapolis, MN*, pages 419–429, 1994.

[30] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *Proc. 47th Annual IEEE Annual Symposium on Foundations of Computer Science (FOCS 06)*, pages 563–574, 2006.

[31] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. In *Proc. of the 43rd IEEE Annual Symposium on Foundations of Science (FOCS'02)*, pages 103–112. IEEE Computer Society, 2002.

[32] R. Fischlin and P. Schnorr. Stronger security proofs for RSA and Rabin bits. *J. Cryptology*, 13:221–244, 2000.

[33] G. D. Forney. Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, 12:125–131, 1966.

[34] R.G. Gallager. *Low-Density Parity-Check Codes.* MIT Press, Cambridge, MA, 1963.

[35] A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse fourier representations via sampling. In *Proc. of 34 ACM Annual Symposium on Theory of Computing (STOC'02)*, pages 152–161. ACM Press, 2002.

[36] Anna C. Gilbert, Martin J. Strauss, Joel A. Tropp, and Roman Vershynin. Algorithmic linear dimension reduction in the l_1 norm for sparse vectors. *CoRR*, abs/cs/0608079, 2006.

[37] M. Goldmann and A. Russell. Spectral bounds on general hard core predicates. In *In Proc. of STACS*, pages 614–625, 2000.

[38] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *Proc. 27th ACM Annual Symposium on Theory of Computing (STOC'89)*, pages 25–32, 1989.

[39] O. Goldreich, D. Ron, and M. Sudan. Chinese remaindering with errors. *IEEE Transactions on Information Theory*, 46(4):1330–1338, July 2000.

[40] Oded Goldreich. *Foundations of Cryptography, Basic Tools.* Cambridge University Press, 2001.

[41] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, 28(2):270–299, 1984.

[42] S. Goldwasser, S. Micali, and P. Tong. Why and how to establish a private code on a public network. In *In Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pages 134–144, 1982.

[43] Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. Local decoding and testing for homomorphisms. In *APPROX-RANDOM*, pages 375–385, 2006.

[44] V. Guruswami and P. Indyk. Expander-based constructions of efficiently decodable codes. In *Proc. IEEE Annual Symposium on Foundations of Computer Science (FOCS'01)*, pages 658–667, 2001.

[45] V. Guruswami and P. Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proc. 34th ACM Annual Symposium on Theory of Computing (STOC'02)*, pages 812–821, 2002.

[46] V. Guruswami and P. Indyk. Linear time encodable and list decodable codes. In *Proc. 35th ACM Annual Symposium on Theory of Computing (STOC'03)*, pages 126–135, 2003.

[47] V. Guruswami and A. Rudra. Explicit capacity-achieving list-decodable codes. In *Proc. 38th ACM Annual Symposium on Theory of Computing (STOC'06)*, pages 1–10, 2006.

[48] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *Proc. IEEE Annual Symposium on Foundations of Computer Science (FOCS'98)*, pages 28–39, 1998.

[49] V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. In *Proc. 32nd ACM Annual Symposium on Theory of Computing (STOC'00)*, pages 181–190, 2000.

[50] R.W. Hamming. Error detecting and error correcting codes. Technical journal, Bell System, 1950.

[51] J. Hastad and M. Naslund. The security of individual RSA bits. In *Proc. IEEE Annual Symposium on Foundations of Computer Science (FOCS'98)*, pages 510–521, 1998.

[52] Johan Hastad. Testing of the long code and hardness for clique. In *Proc. of 28th ACM Annual Symposium on Theory of Computing (STOC'96)*, pages 11–19, New York, NY, USA, 1996. ACM Press.

[53] Johan Hastad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[54] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Stat. Assoc*, 58:13–30, 1963.

[55] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. *Lecture Notes in Computer Science*, 2248:52–66, 2001.

[56] R. Impagliazzo. private communication.

[57] J. C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proc. IEEE Annual Symposium on Foundations of Computer Science (FOCS'94)*, pages 42–53, 1994.

[58] G. David Forney Jr. and Mitchell D. Trott. The dynamics of group codes: State spaces, trellis diagrams, and canonical encoders. *IEEE Transactions on Information Theory*, 39(5):1491–1513, 1993.

[59] B. S. Kaliski. A pseudo-random bit generator based on elliptic logarithms. In *In Advances in Cryptology — CRYPTO '86, A. M. Odlyzko, Ed., vol. 263 of Lecture Notes in Computer Science, Springer-Verlag*, volume 263, pages 84–103, 1986.

[60] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proc. 32nd ACM Annual Symposium on Theory of Computing (STOC'00)*, pages 80–86, 2000.

[61] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2 - \varepsilon$. In *IEEE Conference on Computational Complexity*, pages 379–. IEEE Computer Society, 2003.

[62] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SICOMP*, 22(6):1331–1348, 1993.

[63] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

[64] D.L. Long and A. Wigderson. The Discrete Log problem hides $O(\log N)$ bits. *SIAM J. Comp.*, 17:363–372, 1988.

[65] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM J. on Computing*, 24(2):357–368, 1995.

[66] R. J. McEliece. A Public-Key Cryptosystem Based On Algebraic Coding Theory. *Deep Space Network Progress Report*, 44:114–116, January 1978.

[67] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[68] D. E. Muller. Applications of Boolean Algebra to Switching Circuits Design and Error Detection. *IRE Trans. EC*, 3:6–12, September 1954.

[69] M. Naslund. All bits in ax + b modp are hard (extended abstract). In *In Proc. of CRYPTO '96*, pages 114–128, 1996.

[70] Rakesh Agrawal and Christos Faloutsos and Arun N. Swami. Efficient Similarity Search In Sequence Databases. In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, 1993. Springer Verlag.

[71] I. S. Reed. A Class of Multiple-Error-Correcting Codes and the Decoding Scheme. *IRE Tran. IT*, 4(4):38–49, September 1954.

[72] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, June 1960.

[73] O. Regev. New lattice based cryptographic constructions. In *Proc. 35th ACM Annual Symposium on Theory of Computing (STOC'03)*, pages 407–416. ACM Press, 2003.

[74] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proc. 37th ACM Annual Symposium on Theory of Computing (STOC'05)*, pages 84–93, 2005.

[75] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[76] C. E. Shannon. A mathematical theory of communication. Technical journal, Bell System, July and October 1948.

[77] M. A. Shokrollahi and H. Wasserman. List decoding of algebraic-geometric codes. *IEEETIT: IEEE Transactions on Information Theory*, 45, 1999.

[78] M. Sipser and D. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42:1710–1722, 1996.

[79] David Slepian. Group codes for the gaussian channel. Bell System Tech. J. 47, Bell Telephone Laboratories, Murray Hill, NJ, 1968.

[80] D. A. Spielman. Linear-time encodable and decodable error-correcting codes. In *Proc. of 27th ACM Annual Symposium on Theory of Computing (STOC'95)*, pages 388–397, 1995.

[81] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[82] M. Sudan. List Decoding: Algorithms and Applications. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 31(1):16–27, March 2000.

[83] M. Sudan, L. Trevisan, and S. P. Vadhan. Pseudorandom generators without the XOR lemma. *J. of Computer and System Sciences*, 62(2):236–266, 2001.

[84] L. Trevisan. List-Decoding Using The XOR Lemma. In *Proc. 44th Annual Symposium on Foundations of Computer Science (FOCS'03)*, pages 126–135, Cambridge, MA, usa, October 2003. IEEE Computer Society.

[85] L. Trevisan. Some applications of coding theory in computational complexity. *Quaderni di Matematica*, 13:347–424, 2004.

[86] U.V. Vazirani and V.V. Vazirani. Efficient and secure pseudo-random number generation. In *Proc. 25th IEEE Symp. on Foundations of Computer Science*, pages 458–463, 1984.

[87] H. Wee. private communication.

[88] Stephen A. Weis. Security Parallels Between People and Pervasive Devices. In *Pervasive Computing and Communication Security.* IEEE Press, March 2005.

[89] J. M. Wozencraft. List decoding. Quarterly progress report 48, Research Lab. of Electronics, MIT, 1958.

[90] A.C. Yao. Theory and application of trapdoor functions. In *Proc. 23rd IEEE Annual Symposium on Foundations of Computer Science (FOCS'82)*, pages 80–91, 1982.