Finding Significant Fourier Transform Coefficients Deterministically and Locally

Adi Akavia*

November 20, 2008†

Abstract

Computing the Fourier transform is a basic building block used in numerous applications. For data intensive applications, even the $O(N \log N)$ running time of the Fast Fourier Transform (FFT) algorithm may be too slow, and *sub-linear* running time is necessary. Clearly, outputting the entire Fourier transform in sub-linear time is infeasible, nevertheless, in many applications it suffices to find only the τ-*significant Fourier transform coefficients*, that is, the Fourier coefficients whose magnitude is at least τ-fraction (say, 1%) of the energy (*i.e.*, the sum of squared Fourier coefficients). We call algorithms achieving the latter *SFT algorithms*.

In this work we present a *deterministic* algorithm that finds the τ-significant Fourier coefficients of functions f over *any finite abelian group* G in time polynomial in $\log|G|$, $1/\tau$ and $L_1(\widehat{f})$ (for $L_1(\widehat{f})$ denoting the sum of absolute values of the Fourier coefficients of f). Our algorithm is robust to random noise.

Our algorithm is the first deterministic and efficient (*i.e.*, polynomial in $\log|G|$) SFT algorithm to handle functions over any finite abelian groups, as well as the first such algorithm to handle functions over $\mathbb{Z}_N$ that are neither compressible nor Fourier-sparse. Our analysis is the first to show robustness to noise in the context of deterministic SFT algorithms.

Using our SFT algorithm we obtain (1) deterministic (universal and explicit) algorithms for sparse Fourier approximation, compressed sensing and sketching; (2) an algorithm solving the Hidden Number Problem with advice, with cryptographic bit security implications; and (3) an efficient decoding algorithm in the random noise model for polynomial rate variants of Homomorphism codes and any other concentrated & recoverable codes.

*Institute for Advanced Study, Princeton NJ 08540 and DIMACS, Rutgers University, Piscataway, NJ 08854. This research was supported in part by NSF grant CCF-0514167, by NSF grant CCF-0832797, and by Israel Science Foundation 700/08. Email: akavia@ias.edu

†A preliminary version of this work appears in authors PhD dissertation [2].

1 Introduction

Computing the Fourier transform is a basic building block used in numerous algorithms arising in the context of a wide variety of applications. The best known algorithm for computing the entire Fourier transform is the Fast Fourier Transform (FFT) algorithm [14] that computes the Fourier transform in time $O(N \log N)$ for N the input size.

For data intensive applications, even the running time of the FFT algorithm may be too slow, and *sub-linear* running time is necessary. Clearly, achieving sub-linear running time is infeasible when computing the entire Fourier transform, because the output itself is of length N. Nevertheless, in many applications it is not necessary to compute the entire Fourier transform; instead it suffices to find only the τ-*significant Fourier transform coefficients*, that is, the indices and approximate values of the Fourier coefficients whose magnitude is at least τ-fraction (say, 1%) of the sum of squared Fourier coefficients.

In a sequence of works [4, 23, 24, 27, 34–36, 38, 39] starting with the seminal work of Goldreich and Levin [27], it was shown that finding the significant Fourier transform coefficients (aka, *SFT algorithms*) can be done in time polynomial in $\log N$ and $1/\tau$, that is, much much faster than computing the entire Fourier transform.[1] We use the term **efficient** to address SFT algorithms with running time polynomial in $\log N$ and $1/\tau$.

The above algorithms differ on the *domain* of the considered input functions (varying from the boolean cube $\mathbb{F}_2^n$ in [27] to any finite abelian group given by its generators and their orders in [4]), as well as on whether they are *randomized algorithms* [4, 23, 24, 27, 39] or *deterministic ones* [34–36, 38]. Domains addressed by the deterministic algorithms are direct product of groups $\mathbb{Z}_k^n$ for *small modulus* $k = poly(n)$ in [38] and $\mathbb{Z}_N$ in [34–36].

The deterministic algorithms [34–36, 38] are **universal**, that is, they read the *same* set of entries in *all* input functions over the same domain G (provided the same input parameters are given). In addition, they are **explicit**, that is, they choose the set of read entries efficiently and deterministically. In contrast, the randomized algorithms [4,23,24,27,39] are not only non-explicit but also non-universal, that is, they choose fresh entries to be read for each given input function.

Being universal necessitates some sort of restriction on the input function: It is impossible to find the significant Fourier coefficients of *all* functions when reading the same few *fixed* set of entries, because for any function f, changing the few read values to zero has very little affect on the Fourier transform, and yet, clearly the algorithm cannot find the significant Fourier coefficients from reading only those zero values. This issue is addressed in [34–36, 38] by giving *efficient* algorithms only for functions f s.t.:

- $L_1(\widehat{f}) \leq poly \log N$ for $L_1(\widehat{f}) = \sum_\alpha \left|\widehat{f}(\alpha)\right|$ the sum of Fourier coefficient of the input function f and N the domain size in [38].
- f is *p-compressible* for $p \geq 1 + \Omega(1)$, *i.e.*, $\forall b = 1, \ldots, N$ the b-th largest Fourier coefficient of f is of magnitude at most $O(b^{-p})$, in [34, 35].
- f is *m-Fourier sparse* for $m \leq poly \log N$, *i.e.*, f has at most m non-zero Fourier coefficients, in [36].

[1] We remark that [39] only implicitly gives an SFT algorithm, whereas explicitly it addresses interpolation of sparse polynomials. The interpolation algorithm requires evaluating polynomials on increasing powers of 2, resulting with an (implicit) SFT algorithm which is applicable to functions over groups $\mathbb{Z}_N$ only for N a *power of* 2 (or direct products of such groups). Alon-Mansour [5] de-randomized the interpolation algorithm [39] for the case of polynomials over $\mathbb{Z}_N$ for *prime* N. This does not result in a deterministic SFT algorithm as it holds for prime N rather than powers of 2. We suspect nevertheless that [5] could be extended to give a deterministic SFT algorithm for functions over $\mathbb{Z}_N$ where N is a power of 2.

We point out that the best (*i.e.*, weakest) of the above restrictions is the first one (when we assume w.l.o.g. that f is normalized to have $\sum_\alpha \left|\widehat{f}(\alpha)\right|^2 = 1$). In general, [34–36, 38] achieve the following complexity in terms of $L_1(\widehat{f})$, p and m: running time polynomial in $\log N$, $1/\tau$ and $L_1(\widehat{f})$ in [38]; running time polynomial in $\log N$ and $(1/\tau)^{(p+1)/(p-1)}$ in [34, 35]; and running time polynomial in $\log N$ and m in [36].

We use the term **local** to address algorithms with running time polynomial in $\log N$, $1/\tau$ and $L_1(\widehat{f})$. The algorithm of [38] is local, whereas the algorithms of [34–36] are not local.

1.1 New Results

Our main result in this paper is a *deterministic*, *local* and *robust* SFT algorithm for functions over *any finite abelian group G*.

> **Main result: Deterministic local SFT algorithm.** There is a deterministic (universal and explicit) algorithm that, given any finite abelian group G (by its generators and their orders), a significance parameter $\tau \in (0,1]$, a bound $t > 0$, and oracle access to a complex-valued function $f\colon G \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \leq t$, outputs all τ-significant Fourier coefficients of f in running time and query complexity polynomial in $\log|G|$, $1/\tau$ and t.

In particular for $t = L_1(\widehat{f})$ the complexity of our algorithm is polynomial in $\log|G|$, $1/\tau$ and $L_1(\widehat{f})$.

Robustness. Our SFT algorithm succeeds also in the presence of random noise. That is, with probability at least 0.99 over the noise, the algorithm outputs the significant Fourier coefficients of f even when given oracle access only to a corrupted version $f' = f + \eta$ for η random noise of parameter $O(\tau)$ sufficiently small; where we say that η *random noise of parameter* ε if entries of η are drawn independently at random from distributions of expected absolute values at most ε. We remark that clearly the SFT algorithm also handles *adversarial noise* s.t. $L_1(\widehat{\eta}) \leq t$.[2]

Our result improves on other deterministic SFT algorithms [34–36, 38] in giving:

1. The first efficient deterministic SFT algorithm for functions over *arbitrary finite abelian groups G*. In comparison, other deterministic algorithms apply to functions over $\mathbb{Z}_k^n$ for small modulus $k = poly(n)$ [38], or over $\mathbb{Z}_N$ [34–36] where the latter is further restricted to handle only compressible or Fourier sparse functions.

   Handling functions over *any finite abelian group* is motivated by the wide range of domains arising in applications, such as: 1-dimensional functions for audio processing, 2- and 3-dimensional functions for image and video processing and multi-dimensional functions for processing feature spaces arising in machine learning applications, *i.e.*, domains $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ with $k = 1, 2, 3$ and large k.

2. The first *local* deterministic SFT algorithm for functions over $\mathbb{Z}_N$ that handles *any* function f in running time polynomial in $\log N$, $1/\tau$ and $L_1(\widehat{f})$. In particular, our algorithm *efficiently* handles the class of all functions f s.t. $L_1(\widehat{f}) \leq poly\log N$. This class of functions is strictly larger than the previously handled functions, as $L_1(\widehat{f}) = O(1)$ for the $(1+\Omega(1))$-compressible functions in [34, 35],

[2] Looking ahead, in Section 8 we present the *Robust SFT* algorithm that handles adversarial noise η s.t. $\|\eta\|_2^2 = O(\tau)$ in query complexity polynomial in $\log|G|$, $1/\tau$ and $L_1(\widehat{f})$, and in running time *sub-linear* in the domain size.

and $L_1(\widehat{f}) \leq poly\log N$ for the $poly\log N$-Fourier sparse functions in [36] (where we assume w.l.o.g that functions are normalized to have unit energy $\sum_\alpha \left|\widehat{f}(\alpha)\right|^2 = 1$).[3]

Handling this wider class of functions is motivated both by the complexity theoretic goal of determining the limits of de-randomization, as well as by natural families of functions arising in data intensive applications having $L_1(\widehat{f}) = poly(\log|G|)$, *e.g.*, poly-log depth decision trees and decision lists (*c.f.* [38]).

3. The first analysis showing robustness to noise in the context of universal SFT algorithms.

   We point out that when it comes to handling noise there is a vast difference between randomized and universal algorithms: For the *randomized* algorithms [4, 23, 24, 27, 39], being robust to noise (even adversarial noise) is fairly straightforward: when run with parameter $\tau' = (\tau - \varepsilon)$ instead of τ, the randomized algorithms find all the significant Fourier coefficients of f even when given access only to its corrupted version $f' = f + \eta$ s.t. $\|\eta\|_2^2 \leq \varepsilon$. In contrast, for the *universal* algorithms [34–36, 38] even random noise is out of scope, as the corrupted function $f' = f + \eta$ typically has very large $L_1(\widehat{f'}) \approx \sqrt{N}$ even if $L_1(\widehat{f})$ was bounded, and f' is typically not Fourier sparse even if f was Fourier sparse.

Using our SFT algorithm we obtain: (1) an algorithm for decoding polynomial rate variants of homomorphism and concentrated & recoverable codes; (2) an algorithm solving the Hidden Number Problems (HNP) with advice with cryptographic bit security implications; and (3) deterministic algorithms for sparse approximation, compressed sensing and sketching; details follow. The determinism/universality of our SFT algorithm is essential for all those results.

1.1.1 New: Decoding polynomial rate concentrated & recoverable codes

An error correcting code $\mathcal{C}$ is a collection of *codewords* C_m encoding *messages* m in a redundant way to allow *decoding*, that is, recovery of the message even in the presence of noise. *Homomorphism codes* $(G,\mathbb{C})$-*Hom* encode messages m in a finite abelian group G by the truth table of the character $\chi_m \colon G \to \mathbb{C}$ corresponding to m (where the characters are all homomorphisms from G to the complex unit sphere, and they correspond to elements $m \in G$ according to an isomorphism between G and the group of its characters). For example, the well know Hadamard code is the homomorphism code over the boolean cube $G = \mathbb{F}_2^n$. In homomorphism codes $(G,\mathbb{C})$-Hom, the codewords length is $|G|$ which is *exponential* in the information rate $\log|G|$ (*i.e.*, in the messages binary representation length).

We ask whether there are restrictions of the codewords of $(G,\mathbb{C})$-Hom to a small subset of their entries yielding new *efficiently decodable* codes of codewords length *polynomial* in the information rate. For homomorphism codes over some *small groups* such restrictions are known; for example, the Sipser-Spielman codes [44] can be viewed as a restriction of the Hadamard codes achieving linear codewords length and efficient decoding in the adversarial noise model. In contrast, for homomorphism codes over *large groups*, *e.g.*, $G = \mathbb{Z}_N$, such restrictions are not known.

In this paper we show that *for every* homomorphism code $(G,\mathbb{C})$-Hom with G a finite abelian group, there is an explicit subset S of the entries of its codewords such that restricting all codewords to the entries in S yields a new code that achieves: (1) codewords length *polynomial* in the information rate $\log|G|$, and (2)

[3] This is without loss of generality because an "approximate normalization" achieving $\|f\|_2^2 \approx 1$ can be computed deterministically by dividing each value of f by an estimate $\frac{1}{|A|}\sum_{x\in A}|f(x)|^2$ for A a small biased set in the domain, and the discussed algorithms can be made oblivious to the distinction between exact vs. approximate normalization.

efficient decoding in the random noise model. Furthermore, we show that such restrictions exist for all codes in the class of *concentrated and recoverable codes*. (The class of concentrated and recoverable codes was introduced in [4] and includes in particular all homomorphism code $(G,\mathbb{C})$-Hom as well as boolean-ization of homomorphism codes called Multiplication Codes [4].)

Our SFT algorithm is a central component in our decoding algorithm for these new polynomial encoding length codes. The universality, determinism, robustness and locality of our SFT algorithm is crucial for this result: Universality is essential for the construction of the new codes, specifically, for defining the restriction S. Determinism enables us to obtain explicit code construction. Robustness allows us to decode in the presence of random noise. Locality allows us to decode not only homomorphism codes (where codewords have a constant sum of Fourier coefficients $L_1(\widehat{C}) = 1$) but also any concentrated and recoverable codes (where the sum of Fourier coefficients is only required to be polynomial in $\log|G|$).

Related prior works. Local testing of homomorphism codes with constant query complexity was given in [7]. Local list decoding of homomorphism codes in poly-logarithmic query and time complexity were given for the Hadamard codes [27], for homomorphism codes $(G,\mathbb{C})$-Hom [4], and for arbitrary homomorphism codes [20, 28]. We remark that [4] and [20] differ also in the considered distance model: The algorithm of [4] decodes from noise approaching the code ℓ_2^2-distance. In particular, for codes accepting values on the complex unit sphere –as are homomorphism codes $(G,\mathbb{C})$-Hom– this implies decoding from noise approaching normalized Hamming distance half. The algorithm of [20] decode from noise approaching the code normalized Hamming distance.[4]

1.1.2 New: Solving Hidden Number Problem with advice, and bit security implications

The *Hidden Number Problem* was formalized by Boneh and Venkatesan [8] in the context of presenting the best known result on the bit security of the Diffie-Hellman function. A relaxation of this problem to a Hidden Number Problem *with advice* was subsequently formalized in the context of a security proof for cryptographic functions such as Okamoto conference key sharing scheme and a modified ElGamal's public key encryption scheme [9].

In the Hidden Number Problem (HNP) with advice, for p a large prime and g a generator of the multiplicative group $\mathbb{Z}_p^*$, the goal is to find a hidden number $s \in \mathbb{Z}_p^*$ when given a *short advice string* that depends only on p and g and oracle access to the function $P_s(a) = MSB_k(s \cdot g^a \bmod p)$ mapping $a \in \{1,\ldots,p\}$ to the k most significant bits in the binary representation of $s \cdot g^a \bmod p$.

Boneh and Venkatesan [9] gave an algorithm solving the HNP with advice for any $k \geq O(\log\log p)$ in running time polynomial in $\log p$. They then use this algorithm to show that computing the value of the k most significant bits is as hard as breaking the scheme for the Okamoto conference key sharing scheme and for a modified ElGamal's public key encryption scheme (where "as hard" here means that there is a $poly\log p$ time reduction from the latter to the former). This is interpreted as evidence for the security of these k bits (assuming the underlying functions are secure).

In this paper, we give an algorithm solving the HNP with advice for any $k \geq 1$, and even in the presence of random noise. This improves on prior works [9] in (1) handling $k \geq 1$ rather than $k \geq O(\log\log p)$, and (2) being robust to random noise.

We then use this improved algorithm to strengthen the security results of [9] for the Okamoto conference key sharing scheme and their variant of ElGamal's public key encryption scheme: We show that non-uniformly computing even the *single* most significant bit of the aforementioned cryptographic func-

[4] For $\mathcal{C}$ a code with codewords of length n, its ℓ_2^2-distance is $\min_{C,C'\in\mathcal{C}} \|C - C'\|_2^2$, its normalized Hamming distance is $\min_{C,C'\in\mathcal{C}} \frac{1}{n} |\{ i \in [n] \,|\, C(i) \neq C'(i)\}|$.

tions is as hard as breaking these schemes (where "non-uniformly" here means in the presence of advice depending only on g and p).

Our algorithm for HNP with advice builds on our SFT algorithm. The universality, locality and robustness of our SFT algorithm is crucial for this application: Universality allows us to use a single advice for all inputs (with fixed g, p). Locality allows us to capture the MSB function which is neither compressible nor Fourier sparse and yet has bounded $L_1(\widehat{MSB}) = poly\log p$. Tobustness allows us to work with noisy oracles to $P_s()$.

1.1.3 New: Deterministic sparse approximation, compressed sensing & sketching algorithms

We present deterministic (universal and explicit) algorithms for sparse approximation, compressed sensing and sketching achieving:

1. The first *deterministic* algorithm for finding a near optimal m-sparse Fourier approximation for functions over $\mathbb{Z}_N$ (and any finite abelian G) in time polynomial in $\log N$, m/ε and $L_1(\widehat{f})$ (for ε the approximation parameter). The input to the algorithm is N, m, ε, $L_1(\widehat{f})$ and oracle access to f.

   In comparison, other sparse Fourier approximation algorithms are either randomized [4,23,24,27,39]; or deterministic but restricted to either functions over $\mathbb{Z}_k^n$ for $k = poly(n)$ [38], or to compressible functions over $\mathbb{Z}_N$ [34,35], or to Fourier sparse functions over $\mathbb{Z}_N$ [36].

2. A *deterministic* compressed sensing and sketching algorithms for vectors $x \in \mathbb{R}^N$ with number of linear measurements and recovery time polynomial in $\log N$, m/ε and $L_1(x) = \sum_{i=1}^N |x_i|$ (for m the number of non-zero terms in the recovered representation and ε the approximation parameter).

   Similar performance can be derived from the prior deterministic algorithms [5, 38] when identifying the input with sparse Fourier representation [38] or with coefficients vector of a sparse polynomial [5].

   In comparison, other compressed sensing and sketching algorithms either rely on a randomized (non-universal) choice of measurements [4, 10, 13, 15, 17, 21, 23, 24, 39]; or are universal but non-explicit [6, 11, 16, 18, 25, 26, 33, 42]; or are deterministic but with a number of measurements and a recovery time greater than any polynomial in $\log N$ [19, 29, 32] (and further restricted to handle only to sparse inputs [32]); or are deterministic and efficient but restricted only to compressible inputs [16, 34, 35], sparse inputs [36], or to specific input functions (*e.g.*, "bucket histograms") [22]. Our algorithm falls short of some of the aforementioned algorithms in having polynomial rather than linear dependence on m.[5]

Robustness. Our results above hold even if the oracle to f or the linear measurements of x are corrupted by noise η which is either (1) random noise of parameter $O(\varepsilon/m)$, or (2) adversarial noise of bounded L_1 norm: $L_1(\widehat{\eta}) \leq L_1(\widehat{f})$ for the sparse Fourier approximation algorithm, and $L_1(\eta) \leq L_1(x)$ for the compressed sensing and sketching algorithm. In the random noise case, the algorithm succeeds with 0.99 probability over the noise. Furthermore, we present an extension of the above algorithms to handle adversarial noise η s.t. $\|\eta\|_2^2 \leq O(\varepsilon/m)$ in query complexity polynomial in $\log N$, m/ε and $L_1(\widehat{f})$, and in running time *sub-linear* in the domain size N.

[5] A few remarks. Compressed sensing algorithms are implicit in [4, 5, 23, 24, 38, 39] as they preceded the introduction of the compressed sensing paradigm [10, 21]. The algorithms [5, 38, 39] are restricted to input lengths N that are powers of 2 [38, 39] or primes [5]; nevertheless, they can handle any input length N by padding the input with zeros to reach the nearest appropriate length N'. [5] focus on sparse input polynomials; nevertheless inspecting their algorithm shows it also handles non-sparse inputs with complexity depending on their L_1 norm.

In comparison prior works handling comparable amounts of noise have running time polynomial in the domain size N (rather than in $\log N$) [10, 17]. Prior works with sub-linear running time address only *limited amounts of noise*: noise flipping $O(1/\log N)$ fraction of the read entries [17], noise η s.t. $L_1(\eta) \leq O(1/\log m)$ [25], no noise [5, 38].[6]

1.1.4 New Techniques

Our deterministic SFT algorithm builds on the randomized algorithm of [4] while (1) using a new set of read entries, and (2) providing a new analysis relying on combinatorial conditions rather than on probabilistic arguments (a preliminary version of our new analysis appears in the author's dissertation [2]).

To define the set of read entries we introduce a new combinatorial property –*small bias on intervals*– which is a strict generalization of small biased sets to sets that fool uniform distributions on a restricted support. We then construct the set of read entries from sets that are small biased on intervals of sizes 2^ℓ for $\ell = 1, \ldots, \lfloor (\log N) \rfloor$. We prove that our set is universal for all input functions with bounded $L_1(\widehat{f})$ relying on Fourier analysis of the constructed set. This Fourier analysis does not extend to handling functions corrupted by random noise due to their large L_1 values. Instead we prove universality for noisy functions by showing the algorithm behaves similarly on the noisy and non-noisy functions.

We remark that the definition of small bias on intervals may be useful beyond this work. In comparison, other deterministic SFT or compressed sensing algorithms rely on combinatorial properties such as small biased sets [38], K-majority k-strongly selective sets [34,35], Restricted Isometry Property (RIP) [12,19,29], and extractor graphs [32].

An additional contribution of this work is in introducing connections between deterministic SFT algorithms to solving the Hidden Number Problem with advice and to defining and decoding polynomial rate variants of homomorphism codes and concentrated & recoverable codes.

Paper Organization

The rest of this paper is organized as follows. Some preliminaries are given in section 2. Our SFT algorithm for functions over $\mathbb{Z}_N$ and its analysis are presented in section 3; see section 4 for the case of functions over arbitrary finite abelian groups. Our results in error correcting codes, in cryptographic bit security, and in sparse approximation/compressed sending/sketching appear in sections 5-7. The extension of our algorithm to handling adversarial noise is outlined in section 8.

2 Preliminaries

In this section we summarize some preliminary terminology, notations and theorems.

Inner product, norms, convolution. The *inner product* of complex valued functions f, g over a domain G is $\langle f, g \rangle \stackrel{def}{=} \frac{1}{|G|} \sum_{x \in G} f(x)\overline{g(x)}$. We denote the *normalized* ℓ_2 *norm* and the ℓ_∞ *norm* of f by $\|f\|_2 \stackrel{def}{=} \sqrt{\langle f, f \rangle}$ and $\|f\|_\infty \stackrel{def}{=} \max\{|f(x)| \,|\, x \in G\}$, and denote the *un-normalized* L_1*-norm* by $L_1(f) = \sum_{x \in G} |f(x)|$. The *convolution* of f and g is the function $f * g \colon G \to \mathbb{C}$ defined by $f * g(x) \stackrel{def}{=} \frac{1}{|G|} \sum_{y \in G} f(y)\overline{g(x-y)}$.

[6] We remark that in some prior works a weaker notion of noise was considered, where an input is called "noisy" if it is not sparse. The notion considered here is stronger: we address non-sparse inputs with the additional noise incurred by inaccurate measurements.

Characters and Fourier transform. We denote by $\mathbb{Z}_N \stackrel{def}{=} \mathbb{Z}/N\mathbb{Z}$ the additive group of integers modulo N. The *characters* of $\mathbb{Z}_N$ are the functions $\{\chi_\alpha \colon \mathbb{Z}_N \to \mathbb{C}\}_{\alpha\in\mathbb{Z}_N}$ defined by $\chi_\alpha(x) \stackrel{def}{=} \omega_N^{\alpha x}$ for $\omega_N = e^{2\pi i/N}$ a complex N-th root of unity. For arbitrary finite abelian groups G, the characters are the set of all homomorphism $\chi \colon G \to \mathbb{C}$ from G into the complex unit sphere. The *Fourier transform* of a complex valued function f over G is the function $\widehat{f} \colon G \to \mathbb{C}$ defined by $\widehat{f}(\alpha) \stackrel{def}{=} \langle f, \chi_\alpha\rangle$. A few useful properties: *Parseval Identity* says that $\|f\|_2^2 = \sum_\alpha \left|\widehat{f}(\alpha)\right|^2$. By the *convolution-multiplication duality*, $\widehat{f * g}(\alpha) = \widehat{f}(\alpha)\cdot\widehat{f}(\alpha)$. The Fourier coefficients for the function $g = f\cdot\chi_{-\alpha_0}$ are $\widehat{g}(\alpha) = \widehat{f}(\alpha-\alpha_0)$ (where subtraction is modulo N).

Significant Fourier coefficients. For any $\alpha \in \mathbb{Z}_N$, $val_\alpha \in \mathbb{C}$ and $\tau, \varepsilon \in [0,1]$, we say that α is a τ-*significant Fourier coefficient* iff $\left|\widehat{f}(\alpha)\right|^2 \geq \tau\|f\|_2^2$, and 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 $\mathsf{Heavy}_\tau(f)$.

Small biased sets [41]. We say that a set $A \subseteq \mathbb{Z}_N$ is γ-*biased in* $\mathbb{Z}_N$ if $|\mathbb{E}_{x\in A}[\chi_\alpha(x)]| \leq \gamma$ for every non trivial character χ_α of the group $\mathbb{Z}_N$, $\alpha \neq 0$.

Fact 1 ([1, 37, 43])**.** *There exists a deterministic algorithm that, given any integer $N > 0$ and real $\gamma > 0$, outputs a set $A \subseteq [0..N-1]$ that is γ-biased in $\mathbb{Z}_N$. The size $|A|$ and the running time are poly$(\log N, 1/\gamma)$.*

New definition: (γ, I)-bias in G. For any abelian group G and subsets $B, I \subseteq G$, we say that B is (γ, I)-*biased in G* if for every character χ of the group G, $|\mathbb{E}_{x\in B\cap I}[\chi(x)] - \mathbb{E}_{x\in I}[\chi(x)]| \leq \gamma$.

Fact 2 ([3])**.** *There exists a deterministic algorithm that, given any integers $0 < M < N$ and real $\gamma > 0$, outputs a set $B \subseteq [0..M]$ that is $(\gamma, [0..M])$-biased in $\mathbb{Z}_N$. The size $|B|$ and the running time are poly$(\log N, 1/\gamma)$.*

Remark. The *construction* of [3] is simple given Fact 1, as [3] show that any γ'-small biased sets in $\mathbb{Z}_M$ for sufficiently small $\gamma' = poly(\gamma, 1/\log N)$ is $(\gamma, [0..M])$-biased in $\mathbb{Z}_N$. Their proof is the main novelty in [3].

Tail inequality. Chernoff/Hoeffding theorem bounds the deviation of a sum of independent random variables from its expectation:

Theorem 3 (Chernoff/Hoeffding Bound [30])**.** *Let $X_1, \ldots, X_t$ be independent random variables of expectations $\mu_1, \ldots, \mu_y$ and bounded values $|X_i| \leq M$. Then, $\forall \eta > 0$, $\Pr[\left|\frac{1}{t}\sum_{i=1}^t X_i - \frac{1}{t}\sum_{i=1}^t \mu_i\right| \geq \eta] \leq 2\cdot exp\left(-\frac{2t\eta^2}{M^2}\right)$.*

Characters average over intervals. Denote by $S_t(\alpha) = \frac{1}{t}\sum_{x=0}^{t-1}\chi_\alpha(x)$ the average value of the character χ_α of $\mathbb{Z}_N$ over an interval $[1..t]$, $t < N$. Then $S_t(\alpha)$ decrease fast with the growth of α (*c.f.*proof in [4]):

Proposition 4. $\forall \alpha \in \mathbb{Z}_N$, $|S_t(\alpha)| < \sqrt{\frac{2}{3}}\left(\frac{N/t}{\mathsf{abs}(\alpha)}\right)$ *for* $\mathsf{abs}(\alpha) = \min\{\alpha, N-\alpha\}$.

3 Finding Significant Fourier Coefficients Deterministically and Locally

In this section we present our algorithm for finding significant Fourier coefficients and its analysis. We focus here on the case of functions over $\mathbb{Z}_N$; see section 4 for the case of arbitrary finite abelian groups G.

Our algorithm is composed of two parts: (1) *Queries generating* part, where a set of entries $S = S(G,\tau,t) \subseteq G$ is chosen, given G, τ and t, and (2) *Fixed queries* part, where the significant Fourier coefficients of a function $f \colon G \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \leq t$ are found, given G, τ and the restriction to S of f (or are found with high probability given the restriction to S of f' a corruption of f by random noise).

3.1 Queries Generating

Our queries generating algorithm constructs the set of entries S using sets that are small biased on intervals $[0..2^\ell]$ for $\ell = 0, \ldots, \log N$ (*c.f.* Fact 2 and the preceding definition in section 2):

Algorithm 5. `Queries Generating`. *Given any positive integer N and positive reals τ and t, output a set $S = \bigcup_{\ell=1}^{\lfloor(\log N)\rfloor}(A - B_\ell)$ for $A, B_1, \ldots, B_{\lfloor(\log N)\rfloor}$ each of size polynomial in $\log N$ and $1/\gamma$ for $\gamma = O(\tau/t^2(1 + \log N))$ sufficiently small s.t.*

- *A is γ-biased in $\mathbb{Z}_N$*
- *B_ℓ is $(\gamma, [0..2^\ell])$-biased in $\mathbb{Z}_N$ for $\ell = 1, \ldots, \lfloor(\log N)\rfloor$*

The sets $A, B_1, \ldots, N_{\lfloor(\log N)\rfloor}$ are constructed deterministically in time polynomial in $\log N$ and $1/\gamma$ using the algorithms guaranteed in Facts 1-2, section 2. We remark that $A - B_\ell$ is the difference set $\{a - b \mid a \in A, b \in B_\ell\}$.

Remark 6. *To obtain a* universal *(albeit, non explicit) SFT algorithm it suffices to give a randomized algorithm generating a set of queries $S = \bigcup_{\ell=1}^{\lfloor(\log N)\rfloor}(A - B_\ell)$ for $A, B_1, \ldots, B_{\lfloor(\log N)\rfloor}$ satisfying the properties in Algorithm 5. A randomized algorithm that outputs such a set S with constant success probability is the algorithm that chooses sets $A \subseteq \mathbb{Z}_N$ and $B_\ell \subseteq [0..2^\ell]$ each of size $O((\log N)(\log\log N)/\gamma^2)$ uniformly at random, and outputs $S = \cup_{\ell=1}^{\lfloor(\log N)\rfloor}(A - B_\ell)$. The size of the resulting set is $|S| = O((\log N)^7 \cdot (t/\tau)^4)$; and in particular, $|S| = O((\log^7 N)/\tau^4)$ for compressible functions (as for such functions $t = L_1(\widehat{f})$ is a constant).* Verifying *that a set S satisfies the properties from Algorithm 5 can be done in quasi-linear time $O(|S| \cdot N)$.*

3.2 Fixed Queries SFT

We give an overview of the fixed queries (`FQ-SFT`) part of our algorithm. At a high level, the `FQ-SFT` is a binary search algorithm that repeatedly:

1. **Partitions** the set of potentially significant Fourier coefficients into two halves.
2. **Tests** each half to decide if it (potentially) contains a significant Fourier coefficient. This is done by **estimating** whether the sum of squared Fourier coefficients in each half exceeds the significance threshold τ.
3. **Continues recursively** on any half found to (potentially) contain significant Fourier coefficients.

At each step of this search, the set of potentially significant Fourier coefficients is maintained as a collection $\mathcal{J}$ of intervals: At the first step of the search, all Fourier coefficients are potentially significant, so $\mathcal{J}$ contains the single interval $J = [1..N]$. At each following search step, every interval $J \in \mathcal{J}$ is partitioned into two sub-intervals J_1 and J_2 containing the lower and upper halves of J respectively, and the set $\mathcal{J}$ is updated to hold only the sub-intervals that pass the test, *i.e.*, those that (potentially) contain a significant Fourier coefficient. After $\log N$ steps this search terminates with a collection $\mathcal{J}$ of length one intervals revealing the frequencies of the significant Fourier coefficients. For all frequencies α of the significant Fourier coefficients, we then compute as an $O(\tau)$-approximation for $\widehat{f}(\alpha)$ the value $val_\alpha = \frac{1}{|A|}\sum_{x \in A-y} f(x)\overline{\chi_\alpha(x)}$ for some arbitrary $y \in \cup_{\ell=1}^{\lfloor(\log N)\rfloor} B_\ell$; to simplify notations in the following we assume w.l.o.g. that $y = 0$.

The heart of the algorithm is the test deciding which intervals potentially contain a significant Fourier coefficient (aka, distinguishing procedure). The distinguishing procedure we present, given an interval

J, answers YES if its Fourier weight $weight(J) = \sum_{\alpha \in J} \left|\widehat{f}(\alpha)\right|^2$ exceed the significance threshold τ, and answers NO if the Fourier weight of a slightly larger interval $J' \supseteq J$ is less than $\tau/2$. This is achieved by estimating the ℓ_2 norm (*i.e.*, sum of squared Fourier coefficients) of a filtered version of the input function f, when using a filter h that passes Fourier coefficients in J and decays fast outside of J.

The filters h that we use for depth ℓ of the search are the (normalized) *periodic square function* of support size 2^ℓ or Fourier domain translations of this function:

$$h_{\ell,c}(y) = \begin{cases} \frac{N}{2^\ell} \cdot \chi_{-c}(y) & y \in [0..2^\ell] \\ \\ 0 & otherwise \end{cases}$$

The filter $h = h_{\ell,c}$ passes all frequencies that lie within the length $N/2^\ell$ interval J centered around c, and decays fast outside of J. The filtered version of f is $f * h$, and we estimate its ℓ_2 norm $\|f*h\|_2^2$ by the estimator:

$$\mathsf{est}_{h,A,B_\ell}(f) = \frac{1}{|A|} \sum_{x \in A} \left(\frac{1}{|B_\ell|} \sum_{y \in B_\ell} \chi_{-c}(y) f(x-y) \right)^2$$

for $A, B_1, \ldots, B_\ell \subseteq \mathbb{Z}_N$ as specified in the Queries Generating Algorithm 5.

A pseudo-code of the `FQ-SFT` algorithm follows; we denote by $\{a,b\}$ the interval $[a..b]$ and by $Candidate_\ell$ the collection $\mathcal{J}$ as reached at search depth ℓ.

Algorithm 7. FQ-SFT Algorithm
Input: $N \in \mathbb{N}$, $\tau \in (0,1]$, $A, B_1, \ldots, B_{\log N} \subseteq \mathbb{Z}_N$ *and* $\{(x, f(x))\}_{x \in S}$ *for* $S = A - \bigcup_{\ell=1}^{\log N} B_\ell$
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 $\{a,b\} \in \left\{ \left\{ a', \frac{a'+b'}{2} \right\}, \left\{ \frac{a'+b'}{2} + 1, b' \right\} \right\}$
      i. *Run Distinguishing Algorithm 8 on input* $\{a,b\}$, $\tau \|f\|_2^2$, $A, B_{\ell+1}$, *and* $\{(x, f(x))\}_{x \in S}$; *denote its output by "decision"*
      ii. *If* $decision = 1$, $Candidate_{\ell+1} \leftarrow Candidate_{\ell+1} \bigcup \{\{a,b\}\}$
3. *Output* $L = \left\{ \alpha \,|\, \{\alpha, \alpha\} \in Candidate_{\log N} \right\}$ *and* $\left\{ val_\alpha = \frac{1}{|A|} \sum_{x \in A} f(x) \chi_\alpha(x) \right\}_{\alpha \in L}$

Algorithm 8. Distinguishing Algorithm.
Input: $\{a,b\} \in \mathbb{Z}_N \times \mathbb{Z}_N$, $\tau \in \mathbb{R}^+$, $A, B \subseteq \mathbb{Z}_N$, $\{(x, f(x))\}_{x \in A-B}$
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* $decision = 0$

We remark that to ease the reading of the above pseudo-code we made the simplifying assumptions that f is normalized to have unit energy $\sum_\alpha \left|\widehat{f}(\alpha)\right|^2 = 1$, that $A \subseteq S$, and that $(a'+b')/2$ is an integer. When this is not the case mild changes are due: When f is not normalized we normalize it by dividing each read value by an estimator for the energy of precision $O(\tau\|f\|_2^2)$ sufficiently small; the estimator we use is $\frac{1}{|A|}\sum_{x\in A} f(x)^2$ for A the small bias set computed in Generate Queries algorithm 5 (this is an estimator for $\|f\|_2^2$ which is equal to the energy by Parseval Identity). When A is not contained in S, we replace the sum on A in computing the values val_α's and the estimator for $\|f\|_2^2$ by a sum on $A-y$ for an arbitrary y in $\bigcup_{\ell=1}^{\log N} B_\ell$. When $(a'+b')/2$ is not an integer, we partition $\{a',\ldots,b'\}$ into two disjoint subintervals $\{a',\ldots,c\},\{c+1,\ldots,b'\}$ of roughly the same length.

3.3 Analysis

In this section we analyze our SFT algorithm proving our main result. Recall that $\mathsf{Heavy}_\tau(f)$ is the set of τ-significant Fourier coefficients of f, and that val_α is an ε-approximation for $\widehat{f}(\alpha)$ iff $\left|val_\alpha - \widehat{f}(\alpha)\right| < \varepsilon$.

The following theorem says that our SFT algorithm succeeds when there's no noise.

Theorem 9. *For every positive integer N, positive reals τ,t, and a complex valued function $f\colon \mathbb{Z}_N \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \le t$, our SFT algorithm given N, τ, t and oracle access to f, outputs a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with $O(\tau)$-approximations for $\widehat{f}(\alpha)$ $\forall \alpha \in L$ in running time polynomial in $\log N$, $1/\tau$ and t.*

Proof. Proof follows from the combination of lemma 11 below together with Item 1 in lemma 12 below. □

The next theorem says that our SFT algorithm succeeds also in the presence of noise, that is, the algorithm outputs the significant Fourier coefficients of f even when given only oracle access to a corrupted version $f' = f+\eta$. The noise η may be either (1) *Random noise* of parameter $\varepsilon = O(\tau)$ sufficiently small, that is, entries of η are drawn independently at random from distributions of expected absolute value at most ε, or (2) *Adversarial noise* s.t. $L_1(\widehat{\eta}) \le t$.

Theorem 10 (Robustness to noise)**.** *For every positive integer N, positive reals τ,t, and complex valued functions $f,\eta\colon \mathbb{Z}_N \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \le t$,*

- *Our SFT algorithm, given N, τ, t and oracle access to $f' = f+\eta$ for η random noise of parameter $O(\tau)$ sufficiently small, outputs a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with $O(\tau)$-approximations for $\widehat{f}(\alpha)$ $\forall \alpha \in L$ with probability at least $1 - 1/N^{\Theta(1)}$ over the noise η.*
- *Our SFT algorithm, given N, τ, t and oracle access to $f' = f+\eta$ for η adversarial noise s.t. $L_1(\widehat{\eta}) \le t$, outputs a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with $O(\tau)$-approximations for $\widehat{f}(\alpha)$ $\forall \alpha \in L$.*

The running time of the SFT algorithm polynomial in $\log N$, $1/\tau$ and t.

Proof. The proof for the case of random noise follows from the combination of lemma 11 below together with Item 2 in lemma 12 below. The proof for the case of adversarial noise η follows from the combination of lemma 11 below together with Item 1 in lemma 12 below when observing that $L_1(\widehat{f'}) \le 2t$ for $f' = f+\eta$ s.t. $L_1(\widehat{f}), L_1(\widehat{\eta}) \le t$ implying that running the algorithm with parameter $2t$ rather t suffices. □

Our main lemmas are stated below; proofs appear in section 3.4. Lemma 11 shows that the `FQ-SFT` algorithm succeed on any function f that satisfies conditions (*) and (*') below.

Lemma 11. *For every function $f\colon \mathbb{Z}_N \to \mathbb{C}$ and thresholds $t,\tau > 0$, the* FQ-SFT *algorithm returns a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with τ-approximations for $\widehat{f}(\alpha)$ $\forall \alpha \in L$ in running time polynomial in $\log N$, $1/\tau$ and t if both the following conditions hold:*

$$(*) \qquad \left|\mathsf{est}_{h,A,B_\ell}(f) - \|f * h\|_2^2\right| < c\tau \qquad \forall \ell \in [\lfloor(\log N)\rfloor], c \in \mathbb{Z}_N \text{ and } h = h_{\ell,c} \text{ as defined above, and}$$

$$(*') \qquad \left|\frac{1}{|A|}\sum_{x\in A} f(x)\overline{\chi_\alpha(x)} - \widehat{f}(\alpha)\right| < c\tau \qquad \forall \alpha \in \mathbb{Z}_N$$

for $c > 0$ a sufficiently small absolute constant.

Lemma 12 shows that when using a set of queries S generated by algorithm 5 conditions (*) and (*') hold in any of the following cases: (1) They hold any function f s.t. $L_1(\widehat{f})$; and (2) they hold with high probability for any function $f' = f + \eta$ s.t. f s.t. $L_1(\widehat{f}) \le t$ and η random noise.

Lemma 12. *Let $S = \bigcup_{\ell=1}^{\log N}(A - B_\ell)$ be the output of the queries generating algorithm 5 then for every $f,\eta\colon \mathbb{Z}_N \to \mathbb{C}$ the following holds:*

1. *If $L_1(\widehat{f}) \le t$, then conditions (*) and (*') hold for f.*
2. *If $L_1(\widehat{f}) \le t$ and η is random noise of parameter $\varepsilon \le O(\tau)$ sufficiently small, then conditions (*) and (*') hold for $f' = f + \eta$ with probability at least $1 - 1/N^{\Theta(1)}$ over the noise η.*

3.4 Proofs of Lemmas 11-12

In this section we give the proofs of our main lemmas, lemmas 11-12. Throughout this section conditions (*) and (*') are as defined in Lemma 11.

3.4.1 Proof of Lemma 11

The following lemma gives a sufficient condition for the success of the FQ-SFT algorithm on any particular input function.

Lemma 11. For every function $f\colon \mathbb{Z}_N \to \mathbb{C}$ and thresholds $t,\tau > 0$, the FQ-SFT algorithm returns a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with τ-approximations for $\widehat{f}(\alpha)$ $\forall \alpha \in L$ in running time polynomial in $\log N$, $1/\tau$ and t if the following conditions hold:

$$(*) \qquad \left|\mathsf{est}_{h,A,B_\ell}(f) - \|f * h\|_2^2\right| < c\tau \qquad \forall \ell \in [\lfloor(\log N)\rfloor], c \in \mathbb{Z}_N \text{ and } h = h_{\ell,c} \text{ as defined above, and}$$

$$(*') \qquad \left|\frac{1}{|A|}\sum_{x\in A} f(x)\overline{\chi_\alpha(x)} - \widehat{f}(\alpha)\right| < c\tau \qquad \forall \alpha \in \mathbb{Z}_N$$

for $c > 0$ a sufficiently small absolute constant.

Proof. The lemma is established by showing that if condition (*) holds then the FQ-SFT algorithm efficiently returns all τ-significant Fourier coefficients of f. The fact that the outputted values $val_\alpha = \sum_{x\in A} f(x)\chi_\alpha(x)$ are $O(\tau)$-approximations for the $\widehat{f}(\alpha)$ follows immediately from condition $(*')$.

Correctness. To establish the correctness of the algorithm it suffices to show that the distinguishing procedure answers YES whenever the considered interval J contains a significant Fourier coefficient, *i.e.*,

$\mathsf{est}_{h,A,B_\ell}(f) \geq \Omega(\tau)$ for the used filter $h = h_{\ell,c}$ (with c, $N/2^{\ell-1}$ the center of J and its length). This is true because when J contains a τ-significant Fourier coefficient, then by Proposition 13 Item (1), $\|f*h\|_2^2 \geq \Omega(\sum_{\alpha\in J}\left|\widehat{f}(\alpha)\right|^2) \geq \Omega(\tau)$, implying by condition (*) that also $\mathsf{est}_{h,A,B_\ell}(f) \geq \Omega(\tau)$, and thus the distinguishing procedure decides YES.

Efficiency. To establish the efficiency of the algorithm it suffices to show that the distinguishing procedure does not answer YES too often. If the distinguishing procedure answers YES on a considered interval J, then $\mathsf{est}_{h,A,B_\ell}(f) \geq \Omega(\tau)$ implying by condition (*) that $\|h*f\|_2^2 \geq \Omega(\tau)$. By Proposition 13 Item (2) the latter implies that for a slightly larger interval $J' \supseteq J$, $|J'|/|J| \leq O(1/\gamma)$, its Fourier weight (that is, sum of squared Fourier coefficients with frequencies in J') is greater than $\Omega(\tau)$. This implies that the distinguishing procedure cannot answer YES too often because there are at most $O(1/\tau)$ disjoint intervals whose Fourier weight exceeds $\Omega(\tau)$ (by Parseval Identity), and thus at most $O(\frac{1}{\tau}\frac{|J'|}{|J|})$ (possibly, overlapping) intervals J' whose Fourier weight exceeds $\Omega(\tau)$. □

For integers $\ell, c > 0$ and real $\gamma > 0$, let $J_{\ell,c} = \left\{\alpha \,|\, \mathsf{abs}(\alpha - c) \leq \frac{N}{2^\ell}\right\}$ be an interval in $[0..N-1]$ and let its extension be $J'_{\ell,c,\gamma} = \left\{\alpha \,|\, \mathsf{abs}(\alpha - c) \leq \sqrt{\frac{2}{3\gamma}}\cdot\frac{N}{2^\ell}\right\}$. Then the following holds:

Proposition 13. *(1)* $\|h_{\ell,c}*f\|_2^2 \geq \frac{1}{6}\sum_{\alpha\in J_{\ell,c}}\left|\widehat{f}(\alpha)\right|^2$, *and (2)* $\|h_{\ell,c}*f\|_2^2 \leq \sum_{\alpha\in J'_{\ell,c,\gamma}}\left|\widehat{f}(\alpha)\right|^2 + \gamma$.

Proof. Let $h = h_{\ell,c}$. We first give some properties of h derived using Fourier analysis. Denote $S_t(\alpha) = \frac{1}{t}\sum_{y=0}^{t-1}\chi_\alpha(y)$, and observe that $\widehat{h}(\alpha) = S_{2^\ell}(\alpha - c)$. By Proposition 14 below this implies the following properties of h: (i) $\forall\alpha$, $\left|\widehat{h}(\alpha)\right|^2 \leq 1$, (ii) $\forall\alpha\in J_{\ell,c}$, $\left|\widehat{h}(\alpha)\right|^2 \geq \Omega(1)$, and (iii) $\sum_{\alpha\notin J_{\ell,c,\gamma}}\left|\widehat{h}(\alpha)\right|^2 \leq \gamma$. Recall also that we assumed w.l.o.g that f is normalized to have (iv) $\sum_\alpha\left|\widehat{f}(\alpha)\right|^2 = 1$, which in particular implies that (v) $\forall\alpha$, $\left|\widehat{f}(\alpha)\right|^2 \leq 1$.

Item (1) of Proposition 13 follows from (ii), because by Parseval Identity and the convolution-multiplication duality, $\|h*f\|_2^2 = \sum_\alpha\left|\widehat{h}(\alpha)\right|^2\left|\widehat{f}(\alpha)\right|^2 \geq \Omega(1)\sum_{\alpha\in J_{\ell,c}}\left|\widehat{f}(\alpha)\right|^2$ (where the last inequality follows from (ii)).

Item (2) of Proposition 13 follows from (i),(iii)-(v), because $\|h*f\|_2^2 \leq \max_\alpha\left|\widehat{h}(\alpha)\right|^2\sum_{\alpha\in J_{\ell,c,\gamma}}\left|\widehat{f}(\alpha)\right|^2 + \max_\alpha\left|\widehat{f}(\alpha)\right|^2\sum_{\alpha\notin J_{\ell,c,\gamma}}\left|\widehat{h}(\alpha)\right|^2 \leq \sum_{\alpha\in J_{\ell,c,\gamma}}\left|\widehat{f}(\alpha)\right|^2 + \gamma$ (where the last inequality follows from (iii)-(v)). □

Proposition 14. *Let* $t \in 1,\ldots,N$, *and* $S_t(\alpha) = \frac{1}{t}\sum_{y=0}^{t-1}\chi_\alpha(y)$, *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. □

3.4.2 Proof of Lemma 12 Item 1

The following lemma shows that when using a set of queries S generated by algorithm 5, conditions (*) and (*') hold for every function f of bounded $L_1(\widehat{f})$.

Lemma 12 Item 1. Let $S = \bigcup_{\ell=1}^{\log N}(A-B_\ell)$ be the output of the queries generating algorithm 5, then for every function $f\colon \mathbb{Z}_N \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \leq t$, conditions (*) and (*') hold.

Proof. We first argue that condition (*) holds. Fix N, $\ell \in [\lfloor(\log N)\rfloor]$, A γ-biased in $\mathbb{Z}_N$, and $B = B_\ell$ (γ, I)-biased in $\mathbb{Z}_N$ for $I = [0..2^\ell]$. Denote $g_x(y) = \chi_{-c}(y)\overline{f(x-y)}$ for $y \in I$ and $g_x(y) = 0$ otherwise. By the definition of $\mathsf{est}_{h,A,B}(f)$ and $\|h*f\|_2^2$,

$$\begin{aligned}
\left|\mathsf{est}_{h,A,B}(f) - \|h*f\|_2^2\right| &= \left|\mathop{\mathbb{E}}_{x\in A}\left(\mathop{\mathbb{E}}_{y\in B} g_x(y)\right)^2 - \mathop{\mathbb{E}}_{x\in\mathbb{Z}_N}\left(\mathop{\mathbb{E}}_{y\in I} g_x(y)\right)^2\right| \leq (i)+(ii) \\
\text{for:} \qquad (i) &= \left|\mathop{\mathbb{E}}_{x\in A}\left(\mathop{\mathbb{E}}_{y\in B} g_x(y)\right)^2 - \mathop{\mathbb{E}}_{x\in A}\left(\mathop{\mathbb{E}}_{y\in I} g_x(y)\right)^2\right| \\
(ii) &= \left|\mathop{\mathbb{E}}_{x\in A}\left(\mathop{\mathbb{E}}_{y\in I} g_x(y)\right)^2 - \mathop{\mathbb{E}}_{x\in\mathbb{Z}_N}\left(\mathop{\mathbb{E}}_{y\in I} g_x(y)\right)^2\right|
\end{aligned}$$

We show below that $(i) \leq \gamma\cdot L_1(\widehat{f})^2\cdot O(\log N)$ and $(ii) \leq \gamma\cdot L_1(\widehat{f})^2$. Combining these bounds we get that

$$\left|\mathsf{est}_{h,A,B}(f) - \|h*f\|_2^2\right| \leq \gamma L_1(\widehat{f})^2(O(\log N)+1)$$

Thus, for $\gamma = O(\frac{\tau}{t^2(1+\log N)})$ sufficiently small, $\left|\mathsf{est}(f) - \|h*f\|_2^2\right| \leq O(\tau)$ for all f s.t. $L_1(\widehat{f}) \leq t$.

We next argue that condition $(*')$ holds. Observe that when switching to the Fourier representation of f, $\frac{1}{|A|}\sum_{x\in A} f(x)\overline{\chi_\alpha(x)}$ is equal to $\widehat{f}(\alpha) + \sum_{\beta\neq\alpha}\widehat{f}(\beta)\frac{1}{|A|}\sum_{x\in A}\chi_{\beta-\alpha}(x)$. So, $\left|\frac{1}{|A|}\sum_{x\in A} f(x)\overline{\chi_\alpha(x)} - \widehat{f}(\alpha)\right|$ is upper

bounded by $\sum_{\beta\neq\alpha}\left|\widehat{f}(\beta)\right|\left|\frac{1}{|A|}\sum_{x\in A}\chi_{\beta-\alpha}(x)\right|$ which is in turn upper bounded by $\gamma L_1(\widehat{f})$ for any γ-biased set A. Finally, this implies condition (*') for the choice of $\gamma < O(\tau/L_1(\widehat{f}))$ in our algorithm.

Bounding term (i). Rewrite (i) as $\left|\mathbb{E}_{x\in A}\left[(\mathbb{E}_{y\in B}g_x(y))^2-(\mathbb{E}_{y\in I}g_x(y))^2\right]\right|$ and observe that the expectation over A is upper bounded by the maximum over A. Namely, denoting $g(y)=g_{x_0}(y)$ for the $x_0\in A$ where the maximum is obtained, we have that

$$(i)\leq\left|\left(\mathop{\mathbb{E}}_{y\in B}g(y)\right)^2-\left(\mathop{\mathbb{E}}_{y\in I}g(y)\right)^2\right|$$

Using the identity $a^2-b^2=(a-b)(a+b)$ and observing that $a+b\leq 2\|f\|_\infty$ for $a=\mathbb{E}_{y\in B}g_x(y)$ and $b=\mathbb{E}_{y\in I}g_x(y)$ (where we use here the fact that $\|\chi_{-c}\|_\infty\leq 1$), we get that:

$$(i)\leq 2\|f\|_\infty\left|\mathop{\mathbb{E}}_{y\in B}g(y)-\mathop{\mathbb{E}}_{y\in I}g(y)\right|$$

Switching to the Fourier representation of g and using the triangle inequality we get that:

$$(i)\quad\leq\quad 2\|f\|_\infty\sum_{\alpha\in\mathbb{Z}_N}|\widehat{g}(\alpha)|\left|\mathop{\mathbb{E}}_{y\in B}\chi_\alpha(y)-\mathop{\mathbb{E}}_{y\in I}\chi_\alpha(y)\right|\leq 2\gamma\|f\|_\infty L_1(\widehat{g})$$

where the last inequality follows from the fact that B is γ-biased on I. Observing that by switching to the Fourier representation of f, $\|f\|_\infty=\max_x\left|\sum_\alpha\widehat{f}(\alpha)\chi_\alpha(x)\right|\leq\sum_\alpha\left|\widehat{f}(\alpha)\right|=L_1(\widehat{f})$, we conclude that

$$(i)\quad\leq\quad 2\gamma\cdot L_1(\widehat{f})\cdot L_1(\widehat{g})$$

We next bound $L_1(\widehat{g})$. Observe that $g(y)=h'(y)\overline{f_y(x)}$ for $h'(y)=\chi_{-c}(y)$ if $y\in I$ and $h'(y)=0$ otherwise, and $f_y(x)=f(x-y)$. By the convolution theorem $\widehat{g}=\widehat{h'}*\widehat{f_y}$, implying that

$$L_1(\widehat{g})\leq L_1(\widehat{h'})\cdot L_1(\widehat{f})$$

where the last inequality follows from the fact that $\sum_\alpha\widehat{h'}*\widehat{f_y}(\alpha)\leq L_1(\widehat{h'})\cdot L_1(\widehat{f_y})$, and that $\left|\widehat{f_y}(\alpha)\right|=\left|\widehat{f}(\alpha)\right|$ for all α. Finally, we compute $L_1(\widehat{h'})$. By Proposition 4,

$$\left|\widehat{h'}(\alpha)\right|=\frac{|I|}{N}\left|\frac{1}{|I|}\sum_{x\in I}\chi_{-c+\alpha}(x)\right|\leq\frac{|I|}{N}\frac{N/|I|}{\mathsf{abs}(\alpha-c)}$$

where $\mathsf{abs}(a)$ denotes $\min\{a,N-a\}$. So,

$$L_1(\widehat{h'})=\sum_\alpha 1/\mathsf{abs}(\alpha-c)=O(\log N)$$

We conclude that

$$L_1(\widehat{g})\leq O(\log N)\cdot L_1(\widehat{f})$$

Combining the above bound on (i) with the bound on $L_1(\widehat{g})$ we conclude that:

$$(i)\leq\gamma\cdot L_1(\widehat{f})^2\cdot O(\log N)$$

Bounding term (ii). Denoting $\bar{g}(x) = \left(\mathbb{E}_{y\in I} g_x(y)\right)^2$, we rewrite (ii) as $\left|\mathbb{E}_{x\in A}\bar{g}(x) - \mathbb{E}_{x\in\mathbb{Z}_N}\bar{g}(x)\right|$. Switching to Fourier representation of $\bar{g}$ and using the triangle inequality we upper bound this expression by:

$$(ii) \quad \leq \quad \sum_{\alpha\in\mathbb{Z}_N} \left|\widehat{\bar{g}}(\alpha)\right| \left| \mathop{\mathbb{E}}_{x\in A} \chi_\alpha(x) - \mathop{\mathbb{E}}_{x\in\mathbb{Z}_N} \chi_\alpha(x)\right| \leq \gamma L_1(\widehat{\bar{g}})$$

where in the last inequality we use the fact that A is γ-biased in $\mathbb{Z}_N$.

We next bound $L_1(\widehat{\bar{g}})$. Observe that $\bar{g} = (h*f)^2$ (since $h*f = \mathbb{E}_{y\in\mathbb{Z}_N}\frac{N}{|I|}\chi_{-c}(y)\overline{f(x-y)} = \mathbb{E}_{y\in I}\chi_{-c}(y)\overline{f(x-y)}$). Therefore,

$$L_1(\widehat{\bar{g}}) \leq L_1(\widehat{h*f})^2$$

where we use the fact that for any function s, $L_1(\widehat{s^2}) \leq L_1(\widehat{s})^2$. Observe further that

$$L_1(\widehat{h*f})^2 \leq L_1(\widehat{f})^2$$

because $\left|\widehat{h*f}(\alpha)\right| = \left|\widehat{h}(\alpha)\right| \cdot \left|\widehat{f}(\alpha)\right| \leq \left|\widehat{f}(\alpha)\right|$, where the last inequality follows since $\left|\widehat{h}(\alpha)\right| \leq 1$ for all α. Combining the above bounds together we conclude that

$$(ii) \leq \gamma L_1(\widehat{f})^2$$

□

3.4.3 Proof of Lemma 12 Item 2

The following lemma addresses the random noise case, and shows that when using a set of queries S generated by algorithm 5, conditions (*) and (*') hold with high probability over the choice of noise η for every function $f' = f + \eta$ s.t. f has bounded $L_1(\widehat{f})$.

Lemma 12 Item 2. Let $S = \bigcup_{\ell=1}^{\log N}(A - B_\ell)$ be as in algorithm 5, then with probability at least $1 - 1/N^{\Theta(1)}$, conditions (*) and (*') hold for all functions $f' = f + \eta$ s.t. $L_1(\widehat{f}) \leq t$ and $\eta\colon \mathbb{Z}_N \to \mathbb{C}$ is random noise of expected absolute value $\varepsilon \leq O(\tau)$ sufficiently small (where the probability is taken over the choice of the noise η).

Proof. We first argue that condition (*) holds. Observe that for $f' = f + \eta$, $\left|\mathsf{est}_{h,A,B_\ell}(f') - \|f*h\|_2^2\right| \leq (i) + (ii) + (iii)$ for:

$$\begin{aligned}
(i) \quad &= \quad \left|\mathsf{est}_{h,A,B_\ell}(f) - \|f*h\|_2^2\right| \\
(ii) \quad &= \quad \left|2\frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B_\ell|}\sum_{y\in B_\ell}\chi_{-c}(y)f(x-y)\right)\left(\frac{1}{|B_\ell|}\sum_{y\in B_\ell}\chi_{-c}(y)\eta(x-y)\right)\right| \\
(iii) \quad &= \quad \left|\mathsf{est}_{h,A,B_\ell}(\eta)\right|
\end{aligned}$$

We bound each of these terms. Term (i) is upper bounded by $O(\tau)$ by lemma 1 above. We show below that for each $\ell = 1,\ldots,\lfloor(\log N)\rfloor$, terms (ii) and (iii) are upper bounded by $O(\varepsilon)$, each with probability at least $1 - 1/N^{\Omega(1)}$ (where the probability is over the choice of random noise η). By union bound, both these bounds hold for *all* $\ell = 1,\ldots,\lfloor(\log N)\rfloor$ with probability at least $1 - 2\log N/N^{\Omega(1)} = 1 - 1/N^{\Omega(1)}$. We conclude that for $\varepsilon = O(\tau)$, $\left|\mathsf{est}_{h,A,B_\ell}(f') - \|f*h\|_2^2\right| \leq O(\tau)$ with probability at least $1 - 1/N^{\Omega(1)}$.

We next argue that condition (*') holds. Since $f' = f + \eta$, then $\frac{1}{|A|}\sum_{x\in A} f'(x)\overline{\chi_\alpha(x)}$ is equal to the sum of two terms: $T_1 = \frac{1}{|A|}\sum_{x\in A} f(x)\overline{\chi_\alpha(x)}$ and $T_2 = \frac{1}{|A|}\sum_{x\in A}\eta(x)\overline{\chi_\alpha(x)}$. By Lemma 1, $\left|T_1 - \widehat{f}(\alpha)\right| \le O(\tau)$. To bound the second term observe that $\mathbb{E}_\eta[T_2] \le \mathbb{E}_\eta[\frac{1}{|A|}\sum_{x\in A}|\eta(x)|] = \varepsilon$, implying by Chernoff bound that $|T_2| \le 2\varepsilon$ with probability at least $1 - \exp(\Omega(|A|\varepsilon^2)) \ge 1 - \frac{1}{N^{\Omega(1)}}$ (where the last inequality follows from the choice of $|A|$ used in our algorithm). Combining both these bounds and assigning $\varepsilon < O(\tau)$ we obtain that $\left|\frac{1}{|A|}\sum_{x\in A} f'(x)\overline{\chi_\alpha(x)} - \widehat{f}(\alpha)\right| \le \left|T_1 - \widehat{f}(\alpha)\right| + |T_2| \le O(\tau)$ *–i.e.*, condition (*') holds– with probability at least $1 - 1/N^{\Omega(1)}$.

Bounding (ii). By Cauchy-Schwartz inequality, $(ii)^2 \le (a)\cdot(b)$ for $(a) = \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B_\ell|}\sum_{y\in B_\ell}\chi_{-c}(y)f(x-y)\right)^2$ and $(b) = \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B_\ell|}\sum_{y\in B_\ell}\chi_{-c}\eta(x-y)\right)^2$. We show below that $|(a)| \le O(1)$. To bound (b), observe that (b) is equal to expression (iii) above, *i.e.*, $(b) = \mathsf{est}_{h,A,B_\ell}(\eta)$; and therefore from the bound on (iii) below we get that $(b) \le O(\varepsilon)$ with probability at least $1 - \exp(\Omega(|A|\varepsilon^2))$. Combining both bounds we conclude that $(ii) \le O(\varepsilon)$.

Bounding (a). Observe that $(a) = \mathsf{est}_{h,A,B_\ell}(f)$ for $h = h_{\ell,c}$, implying by Lemma 1 that $\left|(a) - \|h*f\|_2^2\right| \le \gamma L_1(\widehat{f})^2(1 + O(\log N))$. Next observe that $\|h*f\|_2^2 \le 1$ since $\|h*f\|_2^2 = \sum_\alpha \left|\widehat{h}(\alpha)\widehat{f}(\alpha)\right|^2$ where $\left|\widehat{h}(\alpha)\right| \le 1$ for all α and f is normalized to have $\sum_\alpha \left|\widehat{f}(\alpha)\right|^2 = 1$. We conclude therefore that $|(a)| \le 1 + \gamma L_1(\widehat{f})^2(1 + O(\log N)) = O(1)$ (where the last equality follows from the fact that $\gamma L_1(\widehat{f})^2(1 + O(\log N) \le O(\tau)$ for the γ used in our algorithm, and from the fact that $\tau \le 1$).

Bounding (iii). Recall that $\mathsf{est}(\eta) = \frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B_\ell|}\sum_{y\in B_\ell}\chi_{-c}(y)\eta(x-y)\right)^2$ which is upper bounded by $\frac{1}{|A|}\sum_{x\in A}\left(\frac{1}{|B_\ell|}\sum_{y\in B_\ell}|\eta(x-y)|\right)^2$. In expectation $\mathbb{E}[(iii)] \le \varepsilon^2$. By Chernoff bound, we get $Pr[(iii) > \Omega(\varepsilon)] < \exp(\Omega(|A|\varepsilon^2))$. □

4 Finding Significant Fourier Coefficients over Finite Abelian Groups

In this section we describe our SFT algorithm for the case of functions over arbitrary *finite abelian groups*. Our algorithm is composed of two parts: (1) queries generating and (2) fixed queries SFT; described in sections 4.1-4.2 below. Analysis overview is given in section 4.3.

Theorem 15. *There is a deterministic algorithm that for every finite abelian group G, positive reals τ, t, and a complex-valued function $f\colon G \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \le t$, given G (by its generators and their orders), τ, t and oracle access to f, outputs a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with $O(\tau)$-approximations for $\widehat{f}(\alpha)$ $\forall\alpha \in L$. The running time is polynomial in $\log N$, $1/\tau$ and t.*

Furthermore, the above holds with probability at least $1 - 1/N^{\Theta(1)}$ over the random noise even if the algorithm is given oracle access not to f but to a noisy version $f' = f + \eta$ for $\eta\colon \mathbb{Z}_N \to \mathbb{C}$ whose entries are drawn independently at random from a distribution of expected absolute value $O(\tau)$ sufficiently small.

4.1 Queries Generating

The queries generating algorithm constructs the set set S of entries using sets that are small biased on (sets isomorphic to) rectangles $R'_{t+1,\ell} = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_t} \times \{0,\ldots,2^\ell\} \times \{0\} \times \ldots \times \{0\}$ in G.

Algorithm 16. `Queries Generating`. *Given generators $g_1,\ldots,g_k \in G$, their orders $N_1,,\ldots,N_k$, and positive reals τ,t, output a set $S = \bigcup_{\ell\in[\lfloor(\log N)\rfloor],t\in[k]}(A - B_{t,\ell})$ for A and $B_{t,\ell}$'s each of size $spoly(\log|G|,1/\gamma)$ for $\gamma = poly(1/\log|G|,\tau,1/t)$ sufficiently small s.t.*

- *A is a γ-biased set in G*
- *$B_{t,\ell}$ is $(\gamma,R_{t+1,\ell})$-biased in G for each $t = 1,\ldots,k$, $\ell = 1,\ldots,\log N$, where $R_{t+1,\ell}$ is the set in G isomorphic to $R'_{t+1,\ell} = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_t} \times \{0,\ldots,2^\ell\} \times \{0\} \times \ldots \times \{0\}$, i.e., $R_{t+1,\ell} = \left\{ \prod_{j=1}^k g_j^{x_j} \;\middle|\; (x_1,\ldots,x_k) \in R'_{t+1,\ell} \right\}$.*

The sets A and $B_{t,\ell}$'s are deterministically constructed in time polynomial in $\log|G|$ and $1/\gamma$ using the explicit algorithms guaranteed in Fact 1 in section 2 and in corollary 17 below.

The following is a corollary from Facts 1 and 2.

Corollary 17. *For every finite abelian group G isomorphic to $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$, real $\gamma > 0$, and subset $J \subseteq G$ isomorphic to $J' = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_t} \times I \times \{0\} \times \ldots \times \{0\}$ for an interval $I = [0..M]$ for $M < N_{t+1}$, there exists explicit construction (i.e., by a deterministic algorithm with running time $poly(\log|G|,1/\gamma)$) constructing a set $B \subseteq G$ of size $poly(\log|G|,1/\gamma)$ which is (γ,J)-biased in G.*

4.2 Fixed Queries SFT

4.2.1 The Case $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$

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

The filter function that we use is

$$h_{G,t,\ell,c}(y_1, \ldots, y_k) = \begin{cases} \left(\prod_{i=t+1}^{k} N_i\right) \cdot \chi_{\alpha_1,\ldots,\alpha_{t-1}}(y_1, \ldots, y_{t-1}) \cdot h_{N_t,\ell,c}(y_t) & \text{if } (y_{t+1}, \ldots, y_k) = 0^{k-t} \\ 0 & \text{otherwise} \end{cases}$$

for $h_{N_t,\ell,c}(y_t) = \frac{N_t}{2^\ell} \chi_{N_t,-c}(y_t)$ if $y_t \in [0..2^\ell]$ and $h_{N_t,\ell,c}(y_t) = 0$ otherwise; $\chi_{N_1,\ldots,N_{t-1},\alpha_1,\ldots,\alpha_{t-1}}(y_1, \ldots, y_{t-1}) = \prod_{j=1}^{t-1} e^{i\frac{2\pi}{N_j}\alpha_j y_j}$ a character in the group $\mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$; and $\chi_{N_t,-c}(y_t) = e^{i\frac{2\pi}{N_t}(-c)y_t}$ a character in the group $\mathbb{Z}_{N_t}$. When G and $N_1, \ldots, N_k$ are clear from the context, we often omit their indices.

A pseudo-code of the algorithm follows.

Algorithm 18. Fixed Queries SFT Algorithm
Input: *$N_1, \ldots, N_k \in \mathbb{N}$, $\tau \in (0,1]$, $A, B_{t,\ell} \subseteq \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ $\forall (t,\ell) \in [k] \times [\lfloor (\log N_t) \rfloor]$ and $\{(x, f(x))\}_{x \in S}$ for $S = A - \bigcup_{(t,\ell) \in [k] \times [\lfloor (\log N_t) \rfloor]} B_{t,\ell}$*
Output: *$L \subseteq \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ and $\{val_\alpha\}_{\alpha \in L}$*
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 19 on input α^t, $\{a,b\}$, τ, $A, B_{t,\ell+1}$ 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 19. Distinguishing Algorithm.
Input*:* $\alpha^t \in \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_{t-1}}$, $\{a,b\} \in \mathbb{Z}_{N_t} \times \mathbb{Z}_{N_t}$, $\tau \in (0,1]$, $A,B \subseteq \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_k}$ *and* $\{(x,f(x))\}_{x \in A-B}$.
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* $= 0$

4.2.2 The Case G is Arbitrary Finite Abelian Group

The SFT Algorithm for arbitrary finite abelian groups G is defined by utilizing the isomorphism between G and a direct product group.[7] as follows.

Given a description $\left\{(g_j,N_j)\right\}_{j=1}^k$ 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 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.

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 $\log|G|$.

4.3 Analysis Overview

We give an overview of the SFT algorithm analysis for the case of functions over arbitrary finite abelian groups. The high level structure of this analysis is similar to the one for functions over $\mathbb{Z}_N$.

The following theorem says that our SFT algorithm succeeds when there's no noise.

Theorem 20. *For every finite abelian group G, positive reals* τ, t, *and a complex valued functions* $f\colon G \to \mathbb{C}$ *s.t.* $L_1(\widehat{f}) \leq t$, *our SFT algorithm given G (by its generators and their orders),* τ, t *and oracle access to* f, *outputs a list* $L \supseteq \mathsf{Heavy}_\tau(f)$ *together with* $O(\tau)$*-approximations for* $\widehat{f}(\alpha)$ $\forall \alpha \in L$. *The running time is polynomial in* $\log|G|$, $1/\tau$ *and* t.

Proof. The proof follows from the combination of lemma 22 below together with Item 1 in lemma 23 below. □

[7] 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 $\prod_{j=1}^k g_j^{x_j} \in G$.

The next theorem says that our SFT algorithm succeeds also in the presence of noise, that is, the algorithm outputs the significant Fourier coefficients of f even when given only oracle access to a corrupted version $f' = f + \eta$. The noise η may be either (1) *Random noise* of parameter $\varepsilon = O(\tau)$ sufficiently small, that is, entries of η are drawn independently at random from distributions of expected absolute value at most ε, or (2) *Adversarial noise* s.t. $L_1(\widehat{\eta}) \leq t$.

Theorem 21 (Robustness to noise)**.** *For every finite abelian group G, positive reals τ, t, and complex valued functions $f, \eta \colon G \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \leq t$,*

- *Our SFT algorithm, given G (by its generators and their orders), τ, t and oracle access to $f' = f + \eta$ for η random noise of parameter $O(\tau)$ sufficiently small, outputs a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with $O(\tau)$-approximations for $\widehat{f}(\alpha)$ $\forall \alpha \in L$ with probability at least $1 - 1/|G|^{\Theta(1)}$ over the noise η.*
- *Our SFT algorithm, given G (by its generators and their orders), τ, t and oracle access to $f' = f + \eta$ for η adversarial noise s.t. $L_1(\widehat{\eta}) \leq t$, outputs a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with $O(\tau)$-approximations for $\widehat{f}(\alpha)$ $\forall \alpha \in L$.*

The running time of the SFT algorithm polynomial in $\log|G|$, $1/\tau$ and t.

Proof. The proof for the case of random noise follows from the combination of lemma 22 below together with Item 2 in lemma 23 below. The proof for the case of adversarial noise η follows from the combination of lemma 22 below together with Item 1 in lemma 23 below when observing that $L_1(\widehat{f'}) \leq 2t$ for $f' = f + \eta$ s.t. $L_1(\widehat{f}), L_1(\widehat{\eta}) \leq t$ implying that running the algorithm with parameter $2t$ rather t suffices. □

Our main lemmas are stated below. Lemma 22 shows that the `FQ-SFT` algorithm succeed on any function f that satisfies conditions (*) and (*') below. Lemma 23 shows that when using a set of queries S generated by algorithm 16 conditions (*) and (*') hold in any of the following cases: (1) They hold any function f s.t. $L_1(\widehat{f})$; and (2) they hold with high probability for any function $f' = f + \eta$ s.t. f s.t. $L_1(\widehat{f}) \leq t$ and η random noise. Proofs are similar to proofs of lemmas 11-12; details omitted (see author's dissertation [2], Chapter 3, for proof of Lemma 22).

Lemma 22. *Denote by $N_1, \ldots, N_k$ the orders of the generators in the given generating set for G. For every function $f \colon G \to \mathbb{C}$ and thresholds $t, \tau > 0$, the `FQ-SFT` algorithm returns all the τ-significant Fourier coefficients of f in time polynomial in $\log|G|$, $1/\tau$ and t if the following condition holds:*

$$(**) \qquad \left|\mathsf{est}_{h,A,B_\ell}(f) - \|f * h\|_2^2\right| < c\tau$$
$$\forall \ell \in [\lfloor (\log N_{t+1}) \rfloor], t \in [k-1], c \in \mathbb{Z}_{N_{t+1}} \text{ and } h = h_{G,t,\ell,c} \text{ as defined above, and}$$

$$(**') \qquad \left| \frac{1}{|A|} \sum_{x \in A} f(x)\overline{\chi_\alpha(x)} - \widehat{f}(\alpha) \right| < c\tau \qquad \forall \alpha \in G$$

for $c > 0$ a sufficiently small absolute constant.

Lemma 23. *Let S be the output of the queries generating algorithm 16, then for every $f, \eta \colon G \to \mathbb{C}$ the following holds:*

1. *If $L_1(\widehat{f}) \leq t$, then conditions (*) and (*') hold for f.*
2. *If $L_1(\widehat{f}) \leq t$ and η is random noise of parameter $\varepsilon \leq O(\tau)$ sufficiently small, then conditions (*) and (*') hold for $f' = f + \eta$ with probability at least $1 - 1/N^{\Theta(1)}$ over the noise η.*

5 Decoding Polynomial Rate Concentrated & Recoverable Codes

We show that for every concentrated and recoverable code there is a restriction of the codewords to a subset S of their entries yielding a new code of *polynomial codeword length* which is *efficiently decodable in random noise model*.

A code is *concentrated and recoverable* [4] if (1) messages and codeword entries can be identified with elements in a finite abelian group G, and when identifying codewords with functions over G mapping entries to values the following holds: (2) for every codeword C, $L_1(\widehat{C}) \leq poly\log|G|$, and (3) there is a *recovery algorithm* that given a frequency $\alpha \in G$ and a significance threshold τ, outputs all codewords C whose α Fourier coefficient is τ-significant in running time polynomial in $\log|G|$ and $1/\tau$.

Examples of concentrated and recoverable codes include *Homomorphism codes* $(G,\mathbb{C})$-Hom and *Multiplication (MPC) codes* $\mathcal{C}_N^P$ s.t. $L_1(\widehat{P}) \leq poly(\log N)$. Where the MPC code $\mathcal{C}_N^P$ is the boolean-ization of $(\mathbb{Z}_N,\mathbb{C})$-Hom by a boolean function P, that is, with codewords $C_m = (P(\chi_m(1)),P(\chi_m(2)),\ldots,P(\chi_m(N)))$ encoding messages $m \in \mathbb{Z}_N$ [4].

Recall that the *random noise model of parameter* ε outputs corrupted codewords $C' = C + \eta$ for C the uncorrupted codeword and η a random function whose entries are drawn independently at random from distributions of expectation absolute value ε. The *Binary Symmetric Channel (BSC_ε)* is an example of this random noise model, where C is a binary codeword accepting ± 1 values, and η is a function accepting values in $\{-2,0,2\}$ whose value on each entry i is chosen independently at random to be: $\eta(i) = -2C(i)$ with probability ε, and $\eta(i) = 0$ otherwise.

Notations. Let G a finite abelian group, $S \subseteq G$. Denote by $\mathcal{C}_G$ a code with codewords identified with functions $C\colon G \to \mathbb{C}$; we assume that a generating set of G and the generators orders is given as part of the specification of $\mathcal{C}_G$. Denote by $\mathcal{C}^S$ the code whose codewords are the restrictions to S of the codewords $C \in \mathcal{C}_G$, that is, the codewords are $C^S\colon S \to \mathbb{C}$ defined by $C^S(i) = C(i)\ \forall i \in S$.

Theorem 24. *There is a (explicit) constant $c > 0$ such that for every $\tau > 0$ and every concentrated and recoverable code $\mathcal{C}_G$, there is a subset $S \subseteq G$ such that the restriction code $\mathcal{C}^S$ is a code of* polynomial codewords length *which is* efficiently decodable *in the random noise model of parameter $O(\tau)$; that is, the codeword length and the running time of the decoding algorithm are $(\frac{1}{\tau}\log|G|)^c$.*

Proof. Let S be the output of our Queries Generating algorithm when given G (by its generators and their orders), τ and an upper bound t on $\max_{C\in\mathcal{C}} L_1(\widehat{C})$. We point out that it suffices to take $t = 1$ for the case of Homomorphism codes, as $L_1(\widehat{C}) = 1$ for all their codewords.

Given a corrupted codeword $w\colon S \to \mathbb{C}$, we think of w as a restriction of a corrupted codeword $w'\colon G \to \mathbb{C}$ of the code C. The decoding algorithm is as follows: (1) Apply our SFT algorithm to find a list L of the significant Fourier coefficients of w'; (2) Apply the recovery algorithm on each frequency $\alpha \in L$ to obtain a list L_α of all codewords for which α is a significant Fourier coefficient; (3) Return the codeword $C \in \bigcup_{\alpha\in L} L_\alpha$ with highest agreement with the given corrupted codeword w on the entries in S.

The success of this algorithm follows from our analysis of our SFT algorithm together with the analysis of [4] of concentrated and recoverable codes: By the properties of our SFT algorithm, with high probability step (1) of the above algorithm returns the significant Fourier coefficients of w' even in the presence of random noise. This proves the success of our algorithm since it was shown in [4] that to decode concentrated and recoverable codes it suffices to (1') find the significant Fourier coefficients of the given corrupted codeword w' and then continue as in steps (2)-(3) of the above algorithm.

The efficiency of this algorithm follows from the efficiency of the SFT and the recovery algorithms. □

6 Solving Hidden Number Problem with Advice & Bit Security

We give an algorithm solving the HNP with advice for any $k \geq 1$, and even in the presence of random noise.

Definition 25 (Hidden Number Problem (HNP) with advice [9])**.** *For p a prime and g a generator of the multiplicative group $\mathbb{Z}_p^*$, the goal is to find a hidden number $s \in \mathbb{Z}_p^*$ when given a* short advice string *that depends only on p and g and an oracle access to the function*

$$P_s(a) = MSB_k(s \cdot g^a \bmod p)$$

mapping $a \in \{1,\ldots,p\}$ to the k most significant bits in the binary representation of $s \cdot g^a \bmod p$

Theorem 26. *For any prime p and a generator g, there is an algorithm solving the Hidden Number Problem with advice for any $k \geq 1$. Furthermore, with probability at least $1 - 1/p^{\Theta(1)}$, the algorithm succeeds even in the presence of random noise flipping each entry of the oracle P_s independently at random with probability $\varepsilon = O(1)$ sufficiently small.*

Proof. To prove the theorem we show there are an algorithm and an advice string $Advice_{p,g}$ of length $poly\log p$ s.t. for every secret $s \in \mathbb{Z}_p^*$, given $Advice_{p,g}$ and oracle access to $P_s(a) = MSB_1(s \cdot g^a)$, the algorithm finds s in running time polynomial in $\log p$. Furthermore, we show the algorithm succeeds in finding s with probability at least $1 - 1/p^{\Theta(1)}$ even if the oracle answers are corrupted by random noise flipping each bit with sufficiently small constant probability.

Fix a prime p and a generator g of the multiplicative group $\mathbb{Z}_p^*$. Let $f_s \colon \mathbb{Z}_p \to \{0,1\}$ be the function

$$f_s(x) = MSB_1(s \cdot x)$$

Denote $\tau = \max_{\alpha \neq 0} \left|\widehat{f_s}(\alpha)\right|^2$ and $t = L_1(\widehat{f_s})$. Fourier analysis of f_s shows that $\tau = \Theta(1)$; $t = O(\log p)$; the most significant Fourier coefficient of f_s is located on the frequencies s and $-s$, and furthermore, the latter is true with high probability even in the presence of random noise (see details in [2]).

The *advice* we use is discrete logs in the base g of elements in $S \subseteq \mathbb{Z}_p$, $|S| = poly(\log p, 1/\tau, t)$, the output of the Queries Generating Algorithm 5 on input N, τ, t:

$$Advice_{p,g} = \{DL_{p,g}(x)\}_{x \in S}$$

where $DL_{p,g}(x)$ is the element $a \in \mathbb{Z}_{p-1}$ s.t. $x = g^a \mod p$. By the bounds on τ, t, the advice string is of length $poly\log p$

The algorithm for finding s is as follows: (1) Run our SFT algorithm on input domain size p, the significance parameter τ, the bound t on $L_1(\widehat{f_s})$ and with oracle access to the restriction of f_s to S (that is, to the values $\{f_s(x)\}_{x \in S}$); denote the outputted list of frequency by L. (2) Output the $\alpha \in L$ s.t. $MSB_1(\alpha \cdot x)$ has highest agreement with the restriction of f_s to S. Observe that when running the SFT algorithm we can answer all oracle queries it makes, because $f_s(x) = P_s(DL_{p,g}(x))$ and, due to its universality, the SFT algorithm queries only on $x \in S$ for which the advice provides the discrete log $a = DL_{p,g}(x)$.

We show that the output is indeed the hidden number s: Since the *most* significant Fourier coefficients of f_s are located on the frequencies s and $-s$ (and furthermore this holds with high probability even in the presence of random noise), then $s, -s \in L$; this in turn implies that s is outputted by the algorithm (with high probability over the random noise).

The running time of this algorithm is dominated by the running time of the SFT algorithm which is polynomial in $\log p$, $1/\tau = O(1)$ and $t = O(\log p)$. □

As a corollary we obtain a strengthening of the security results of [9] for the Okamoto conference key sharing scheme and their variant of ElGamal's public key encryption scheme : We show that non-uniformly (*i.e.*, in the presence of advice depending on g, p) computing even the *single* most significant bit of the aforementioned cryptographic functions is as hard as breaking these schemes.

For completeness, we write here the definitions of the Okamoto conference key sharing scheme and the ElGamal public key encryption scheme as given in [9]:

Okamoto conference key sharing scheme. Bob picks r at random and sends to Alice $c = g^r$. Alice picks a random s and sends $y = x^s$ back. Bob computes $y^{r^{-1}} = g^s$ which is the conference key they use. Since the conference key is determined by Alice's bits alone she can distribute the same key to all members of the conference. Cracking this scheme needs computing the function $OK_g(g^{rs}, g^r, mg^{xr}) = m$.

Modified ElGamal public key encryption scheme. Bob picks a random x and publishes $y = g^x$ as his public key. To send a message m to Bob, Alice picks a random r and sends g^r, my^r. Bob can decode the message by computing $my^r/(g^r)^x$. To break the scheme one has to compute the function $EL_g(g^x, g^{xr}, mg^r) = m$.

7 Deterministic Sparse Approximation, Compressed Sensing & Sketching

7.1 Deterministic Sparse Fourier Approximation

We present a deterministic (universal and explicit) and efficient algorithm for sparse Fourier approximation.

Theorem 27 (sparse Fourier approximation)**.** *There exists a deterministic (universal and explicit) algorithm that for every finite abelian group G, integer $m \geq 0$, reals $t, \varepsilon > 0$, and a complex-valued function $f : G \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \leq t$, given G (by its generators and their orders), m, t, ε and oracle access to f, outputs a near optimal m-terms approximation R for f s.t.*

$$\|f - R\|_2^2 \leq (1+\varepsilon)\|f - R_{opt}\|_2^2$$

for R_{opt} the best m-terms approximation of f in the Fourier basis (up to finite precision). The running time and query complexity of this algorithm is polynomial in $\log|G|$, m/ε and t.

Proof Sketch. Our sparse Fourier approximation algorithm follows from our SFT algorithm via known techniques for converting SFT algorithms to algorithms finding sparse Fourier approximation [23, 39] (*c.f.*, [23], Theorem 9). Applying these techniques on our *deterministic* and efficient SFT algorithm results in a *deterministic* and efficient algorithm for sparse Fourier approximation: The complexity analysis follows by observing that in the proof of Theorem 9 in [23] the SFT algorithm is run with significance parameters $\tau = poly(\varepsilon/m)$ and on functions f' for which $L_1(\widehat{f'}) \leq L_1(\widehat{f})$ for f the input functions (where the latter is true as $f' = f - \sum_{\alpha \in \Gamma} val_\alpha \chi_\alpha$ for Γ a subset of the significant Fourier coefficients of size $poly(\log|G|, m/\varepsilon, t)$ and val_α's are approximations of the Fourier coefficients $\widehat{f}(\alpha)$'s). □

Robustness. The above algorithm for sparse Fourier approximation is robust to random noise of expected absolute value at most $O(\varepsilon/m)$ with probability .99 over the noise; in addition it is robust to any *adversarial* noise η s.t. $L_1(\widehat{\eta}) \leq t$. Furthermore, using the extension of our SFT algorithm for adversarial noise settings, we obtain an algorithm for sparse Fourier approximation for the case of $G = \mathbb{Z}_N$ that handles any *adversarial noise* η s.t. $\|\eta\|_2^2 = O(\varepsilon/m)$ in running time *sub-linear* in the domain size N and with query complexity remains as in the above theorem, *i.e.*, poly-logarithmic in N.

7.2 Deterministic Compressed Sensing and Sketching

In recent years there's growing interest in algorithms finding succinct approximate representations of vectors $x \in \mathbb{C}^N$ by short *sketches* $s \in \mathbb{C}^k$ (typically with $k << N$) such that given s one can *recover* a near optimal m-sparse approximation R of x. Such sketches are useful for example in the context of *streaming* algorithms [31, 40] where the data is too large to be represented explicitly, as well as in *compressed sensing* [10, 21] where data acquisition already reads only the values requires for computing the sketch.

Our sparse Fourier approximation algorithm gives sketching and recovery algorithms for vectors x with sparse representation in their *Fourier basis*. The sketch is the (explicit) set of entries read by the algorithm. The recovery algorithm is our algorithm finding sparse Fourier approximation. The sketch length and the running time of the recovery algorithm are polynomial in $\log N$, m/ε and $L_1(\widehat{x})$ (for $L_1(\widehat{x})$ the sum of absolute values of the entries of the Fourier transform of x).

Furthermore, by a change-of-basis we obtain sketching and recovery algorithms for vectors x with sparse representation in the *standard basis* $L_1(x) = \sum_{i=1}^{N} |x_i| \leq poly \log N$ with sketch length and the running time of the recovery algorithm polynomial in $\log N$, m/ε and $L_1(x)$. This is because as noted in [32] (see footnote 2 there), any algorithm for sparse Fourier approximation reading k entries gives an algorithm for sparse approximation in the standard basis making k *linear measurements* computing inner product of x with appropriate rows of the (inverse) Fourier matrix. The running time of the recovery algorithm is not affected by this change of basis.

Theorem 28 (Sparse recovery for compressed sensing and sketching)**.** *There exists two deterministic (explicit and universal) algorithms: (1) A measurement generating algorithm that, given integers $N, m > 0$ and reals $t, \varepsilon > 0$, outputs a measurement matrix $A \in \mathbb{C}^{poly(\log N, m/\varepsilon, t) \times N}$; and (2) A recovery algorithm, given integers $N, m > 0$, reals $t, \varepsilon > 0$ and the measurements Ax for any every vector $x \in \mathbb{C}^N$ s.t. $\sum_{i=1}^{N} |x_i| \leq t$, outputs an m-terms approximation $R \in \mathbb{C}^N$ s.t.*

$$\|x - R\|_2^2 \leq (1+\varepsilon)\|x - R_{opt}\|_2^2$$

for R_{opt} the best m-terms approximation of x in the standard basis. The running time of the both these algorithms is polynomial in $\log N$, m/ε and t. The recovery algorithm is robust to random noise η of parameter $O(\varepsilon/m)$ and to adversarial noise η of $L_1(\eta) \leq t$.

8 Robust SFT: Handling Adversarial Noise in Sub-Linear Time

We present a deterministic (universal and explicit) SFT algorithm that handles *adversarial noise* of bounded $\|\eta\|_2^2$ in *sub-linear* time. We focus here on the case of functions over $\mathbb{Z}_N$; the algorithm extends to functions over arbitrary finite abelian groups, details omitted.

Theorem 29. *There is a deterministic algorithm that for every positive integer N, reals $\tau, t > 0$, and functions $f, f' : \mathbb{Z}_N \to \mathbb{C}$ s.t. $L_1(\widehat{f}) \leq t$, and $\|f' - f\|_2^2 = O(\tau)$, given N, τ and oracle access to f', outputs a list $L \supseteq \mathsf{Heavy}_\tau(f)$ together with $O(\tau)$-approximations for $\widehat{f}(\alpha)$ $\forall \alpha \in L$ in query complexity $q = poly(\log N, 1/\tau, t)$ and in running time $N^{O(\varepsilon/\tau)} \cdot O(q/\tau^{1.5})$.*

We remark that this result extends to other finite abelian groups.

Our robust algorithm is composed of two parts: (1) queries generating and (2) fixed queries SFT. The queries generating part is identical to Algorithm 5. We describe the fixed queries part (`Robust-SFT`) and sketch its analysis.

Overview of the `Robust-SFT` **algorithm.** The high level of the `Robust-SFT` algorithm is similar to that of the `FQ-SFT` algorithm: Both algorithms are binary search algorithms that progress via a sequence of adaptive tests, where tests at depth ℓ in the search tree are designed to decide whether given length $N/2^\ell$ intervals potentially contain significant Fourier coefficients of the input function. These tests are essentially achieved by estimating the Fourier weight of the given interval (that is, the sum of squared Fourier coefficients of the input function over this interval) and checking whether it exceeds a threshold $O(\tau)$. Since the lengths of the considered intervals decrease exponentially with ℓ, the algorithm zooms into the exact location of the significant Fourier coefficients in $\log N$ search depth.

The `Robust-SFT` algorithm differ from the `FQ-SFT` algorithm on how each of these tests is executed. In the `FQ-SFT` algorithm, tests at search tree depth ℓ use only the input function values $f(x)$ on entries x in the small set $A - B_\ell$ out of all entries $S = \bigcup_{\ell=1}^{\log N}(A - B_\ell)$, namely, on only a $1/\log N$-fraction of the entries. This is not robust against adversarial noise, because an adversary corrupting even only this $1/\log N$-fraction of the entries can diverge the entire search away from finding the significant Fourier coefficient (say, by setting the values on these few entries to 0, thus convincing the algorithm that f has no significant Fourier coefficients).

To overcome this weakness of the `FQ-SFT` algorithm, in the `Robust-SFT` algorithm, tests at each search tree depth ℓ (test ℓ, in short) are executed relying on many more entries of the input function. Specifically, each test ℓ is composed of $O(\varepsilon/\tau)\log N$ *sub-tests* enumerated by $\widetilde{\ell} = \ell, \ldots, \ell + O(\varepsilon/\tau)\log N$, where each sub-test $\widetilde{\ell}$ is executed using entries in $A - B_{\widetilde{\ell}}$. The outcome of test ℓ is determined by the *majority vote over all sub-test* $\widetilde{\ell}$.

Each of those sub-tests $\widetilde{\ell}$ operates as follows. In sub-test $\widetilde{\ell}$, entries $A - B_{\widetilde{\ell}}$ are used for estimating the Fourier weight of length $N/2^\ell > N/2^{\widetilde{\ell}}$ intervals as follows: The given length $N/2^\ell$ interval is divided into *sub-intervals* of length $N/2^{\widetilde{\ell}}$ and the Fourier weight of each of these intervals is estimated using $A - B_{\widetilde{\ell}}$ (where the latter is achieved in the same manner as it is done in the `FQ-SFT` algorithm). The Fourier weight of the entire interval is the sum of the Fourier weights of all its parts. (More precisely, instead of taking the sum of Fourier weights, we decide that an interval potentially contains a significant Fourier coefficient if any of the sub-intervals exceeds the appropriate threshold $O(\tau)$.)

Overview of the analysis of the `Robust-SFT` **algorithm.** *Correctness.* The tests of the `Robust-SFT` algorithm are robust, because an adversary flipping at most ε-fraction of the entries in S cannot change the majority vote over all $O(\varepsilon/\tau)\log N$ sub-tests. Thus, despite the noise, each test of the `Robust-SFT` algorithm returns the correct outcome. This implies the success of the entire algorithm similarly to the analysis of the `FQ-SFT` algorithm.

Complexity. The running time of the `Robust-SFT` is dominated by $N^{O(\varepsilon/\tau)}$. This is because the number of sub-intervals in each sub-test $\widetilde{\ell}$ of test ℓ is $(N/2^\ell)/(N/2^{\widetilde{\ell}}) = 2^{\widetilde{\ell}-\ell}$, which is $N^{O(\varepsilon/\tau)}$ when $\widetilde{\ell} = \ell + O(\varepsilon/\tau)\log N$.

We remark that while this running time is far worse than the $poly\log N$ running time of the `FQ-SFT` algorithm, yet, it is far better than the $N \cdot poly(\log N)$ running time of the trivial exhaustive search algorithm.

Acknowledgments.

The author is grateful to Shafi Goldwasser, Piotr Indyk, Vinod Vaikuntanathan, and Avi Wigderson for helpful comments and discussions.

References

[1] M. Ajtai, H. Iwaniec, J. Komlos, J. Pintz, and E. Szemeredi. Constructions of a this set with small fourier coefficients. *Bull. London Math. Soc.*, 22:583–590, 1990.

[2] A. Akavia. *Learning Noisy Characters, Multiplication Codes and Cryptographic Hardcore Predicates*. PhD dissertation; defended Aug 2007, MIT, EECS, Feb 2008.

[3] A. Akavia, N. Alon, V. Guruswami, and A. Wigderson. Explicit Constructions of Sets Fooling Negligible Size Arithmetic Progressions. In preparation. 2008.

[4] A. Akavia, S. Goldwasser, and S. 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.

[5] N. Alon and Y. Mansour. ε-discrepancy sets and their application for interpolation of sparse polynomials. *IPL: Information Processing Letters*, 54, 1995.

[6] R. Berinde, A. C. Gilbert, P. Indyk, H. J. Karloff, and M. J. Strauss. Combining geometry and combinatorics: A unified approach to sparse signal recovery. *CoRR*, abs/0804.4666, 2008.

[7] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[8] 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.

[9] D. Boneh and R. Venkatesan. Rounding in lattices and its cryptographic applications. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1997.

[10] E. J. Candes, J. K. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[11] E. J. Candes, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

[12] E. J. Candes and T. Tao. Decoding by linear programming. *CoRR*, abs/math/0502327, 2005.

[13] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

[14] J.W. Cooley and J.W. Tukey. An algorithm for machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, Apr 1965.

[15] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[16] G. Cormode and S. Muthukrishnan. Towards an algorithmic theory of compressed sensing. 2005.

[17] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In Paola Flocchini and Leszek Gasieniec, editors, *SIROCCO*, volume 4056 of *Lecture Notes in Computer Science*, pages 280–294. Springer, 2006.

[18] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing: Closing the gap between performance and complexity. *CoRR*, abs/0803.0811, 2008.

[19] R. A. DeVore. Deterministic constructions of compressed sensing matrices. *J. Complex.*, 23(4-6):918–925, 2007.

[20] I. Dinur, E. Grigorescu, S. Kopparty, and M. Sudan. Decodability of group homomorphisms beyond the johnson bound. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 275–284, New York, NY, USA, 2008. ACM.

[21] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 42(4):1289–1306, April, 2005.

[22] Sumit Ganguly and Anirban Majumder. Cr-precise: A deterministic summary structure for update data streams. *CoRR*, abs/cs/0609032, 2006.

[23] 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.

[24] A. C. Gilbert, S. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal sparse fourier representation via sampling. In *in Proc. SPIE Wavelets XI*, 2005.

[25] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. Algorithmic linear dimension reduction in the l_1 norm for sparse vectors. *CoRR*, abs/cs/0608079, 2006.

[26] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. One sketch for all: fast algorithms for compressed sensing. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 237–246, New York, NY, USA, 2007. ACM.

[27] 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.

[28] Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. Local decoding and testing for homomorphisms. In *APPROX-RANDOM*, pages 375–385, 2006.

[29] V. Guruswami, J. R. Lee, and A. Razborov. Almost euclidean subspaces of ℓ_1 via expander codes. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 353–362, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[30] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Stat. Assoc*, 58:13–30, 1963.

[31] P. Indyk. Sketching, streaming and sub-linear space algorithms. graduate course notes, available at http://stellar.mit.edu/s/course/6/fa07/6,895/, 2007.

[32] P. Indyk. Explicit constructions for compressed sensing of sparse signals. In *SODA*, pages 30–33, 2008.

[33] P. Indyk and M. Ruzic. Near-optimal sparse recovery in the l_1 norm. volume 49th Annual IEEE Symposium on Foundations of Computer Science, 2008.

[34] M. A. Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. *CoRR*, abs/0708.1211, 2007.

[35] M. A. Iwen. A deterministic sub-linear time sparse fourier algorithm via non-adaptive compressed sensing methods. In *SODA*, pages 20–29, 2008.

[36] M. A. Iwen and C. V. Spencer. Improved bounds for a deterministic sublinear-time sparse fourier algorithm. In *Conference on Information Sciences and Systems (CISS), Princeton, NJ*, 2008.

[37] M. Katz. An estimate for characters sum. *J. AMS*, 2(2):197–200, 1989.

[38] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SICOMP*, 22(6):1331–1348, 1993.

[39] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *SIAM J. on Computing*, 24(2):357–368, 1995.

[40] S. Muthukrishnan. Data streams: Algorithm and applications (invited talk at soda'03). available at http://athos.rutgers.edu/ muthu/stream-1-1.ps, 2003.

[41] J. Naor and M. Naor. Small biased probability spaces: efficient constructions and applications. volume 22nd ACM Symposium on the Theory of Computing, pages 213–223, 1990.

[42] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comp. Harmonic Anal.*, June 2008.

[43] A. Razborov, A. Wigderson, and E. Szemeredi. Constructing small sets that are uniform in arithmetic progressions. *Combinatorics, Probability and Computing*, 2:513–518, 1993.

[44] M. Sipser and D. Spielman. Expander codes. *IEEE Trans. Inform. Theory*, 42:1710–1722, 1996.

A Definitions of the Okamoto and ElGamal schemes

For completeness, we write here the definitions of the Okamoto conference key sharing scheme and the modified ElGamal public key encryption scheme as given in [9]:

Okamoto conference key sharing scheme. Bob picks r at random and sends to Alice $c = g^r$. Alice picks a random s and sends $y = x^s$ back. Bob computes $y^{r^{-1}} = g^s$ which is the conference key they use. Since the conference key is determined by Alice's bits alone she can distribute the same key to all members of the conference. Cracking this scheme needs computing the function $OK_g(g^{rs}, g^r, mg^{xr}) = m$.

Modified ElGamal public key encryption scheme. Bob picks a random x and publishes $y = g^x$ as his public key. To send a message m to Bob, Alice picks a random r and sends g^r, my^r. Bob can decode the message by computing $my^r/(g^r)^x$. To break the scheme one has to compute the function $EL_g(g^x, g^{xr}, mg^r) = m$.