Finding Significant Fourier Coefficients: Clarifications, Simplifications, Applications and Limitations

Steven D. Galbraith, Joel Laity and Barak Shani
Department of Mathematics, University of Auckland, New Zealand

Abstract

Ideas from Fourier analysis have been used in cryptography for three decades. Akavia, Goldwasser and Safra unified some of these ideas to give a complete algorithm that finds significant Fourier coefficients of functions on any finite abelian group. Their algorithm stimulated a lot of interest in the cryptography community, especially in the context of "bit security". This paper attempts to be a friendly and comprehensive guide to the tools and results in this field. The intended readership is cryptographers who have heard about these tools and seek an understanding of their mechanics, and their usefulness and limitations. A compact overview of the algorithm is presented with emphasis on the ideas behind it. We survey some applications of this algorithm, and explain that several results should be taken in the right context. We point out that some of the most important bit security problems are still open. Our original contributions include: an approach to the subject based on modulus switching; a discussion of the limitations on the usefulness of these tools; an answer to an open question about the modular inversion hidden number problem.

Keywords: Significant Fourier transform, Goldreich-Levin algorithm, Kushilevitz-Mansour algorithm, bit security of Diffie-Hellman.

CONTENTS

1. INTRODUCTION

Let G be a finite abelian group. Fourier analysis provides a convenient basis for the space of functions $G \to \mathbb{C}$, namely the characters $\chi : G \to \mathbb{C}$. It follows that any function $f : G \to \mathbb{C}$ can be represented as a linear combination $f(x) = \sum_{\alpha \in G} \widehat{f}(\alpha)\chi_\alpha(x)$, where $\widehat{f}$ is the discrete Fourier transform of f. A standard problem is to approximate a function, up to any error term, using a linear combination of a small number of characters. This is not always possible, but for certain functions (which are called *concentrated*) it is possible. The coefficients in such an approximation are called *significant* Fourier coefficients, as their size is large relative to the function's norm. The simplest example of a concentrated function is a character itself.

A natural computational problem is to compute such an approximation. When doing this one might have a complete description of the function or, as will be the case in this paper, just a small set of values $f(x_i)$. The ability to choose specific x_i's plays a crucial role in the ability to approximate f. Indeed, the main result in this subject is an algorithm that efficiently computes a *sparse* approximation for any concentrated function on any abelian group G, by computing all its significant coefficients (assuming one is able to choose the inputs x_i). Algorithms that compute significant coefficients first appear explicitly in the works of Goldreich-Levin [18] and Kushilevitz-Mansour [25], though some of the main ideas already appear in earlier works. We use the general term *significant Fourier transform* (SFT) to refer to algorithms of this type. The main aim of our paper is to survey some SFT algorithms and their applications in cryptography.

One simple application is the problem of computing a secret vector $\mathbf{s} \in \mathbb{Z}_2^m$ when given access to a function $f(\mathbf{x})$ on $\mathbb{Z}_2^m$ that computes $\langle \mathbf{s}, \mathbf{x} \rangle + e$ where $e \in \{0, 1\}$ is a noise that is zero with probability $p > 1/2$ (this is the original application studied by Goldreich and Levin). There are many approaches to solving this noisy linear algebra problem; one is to define the function $g(\mathbf{x}) = (-1)^{f(\mathbf{x})}$ and to consider Fourier analysis on the additive group $G = \mathbb{Z}_2^m$. One can show (see Example 4) that the coefficient $\widehat{g}(\mathbf{s})$ corresponding to the character $\chi_{\mathbf{s}}(\mathbf{x}) = (-1)^{\langle \mathbf{s}, \mathbf{x} \rangle}$ is a significant Fourier coefficient. Hence, computing the significant Fourier coefficients for this function gives an algorithm to compute the secret vector $\mathbf{s}$. We emphasize that in order to efficiently compute the significant coefficients it is important to be able to choose the queries $\mathbf{x}$ to the function, otherwise this problem is the learning parity with noise (LPN) problem (also closely related to decoding a random binary linear code) which is believed to be hard.

In some other applications there is a known function f, and we also have access to the function $f_s := f \circ \varphi_s$ for some function φ parameterized by an unknown value s. If f has a significant Fourier coefficient, and if there is some relation between the Fourier transforms $\widehat{f}, \widehat{f_s}$ of f and f_s, then one could hope that this relation would disclose some information on s. For example, taking G to be the additive group $\mathbb{Z}_p$ and $\varphi_s(x) = sx \pmod{p}$, we have the *scaling property* $\widehat{f_s}(\alpha) = \widehat{f}(\alpha s^{-1})$ for every

$\alpha \in G$. It follows that f and f_s share the same coefficients in different order. If α is a significant Fourier coefficient of f and β is a significant Fourier coefficient of f_s then $\alpha\beta^{-1}$ is a candidate value for s. This idea has been used by Akavia, Goldwasser and Safra [3] to give a new approach for the chosen multiplier hidden number problem in $\mathbb{Z}_p^*$, which subsequently led to a new approach in study the security of Diffie-Hellman schemes.

This paper is aimed at the cryptographer who has heard about these tools and seeks a clear understanding of their mechanics, and a framework for their usefulness and their limitations. We describe the SFT algorithm from a high level point of view, stressing the mathematical ideas behind it and the situations in which it can be applied. We study, from the Fourier analysis point of view, different approaches for applying these tools and of proving results in this area. We also show some limitations on applications that use these tools. Moreover, we survey some of the recent results and applications using these tools.

The SFT algorithm and variants have received great attention in the literature outside the regime of cryptography. The Kushilevitz-Mansour algorithm [25] is a cornerstone in this research field, and serves as a basis for most algorithms, including the one we present in this paper. Researchers in engineering, concerned with practical applications in signal processing, have developed algorithms with greater efficiency (with respect to various metrics); for a recent survey on these algorithms see Gilbert, Indyk, Iwen and Schmidt [17].

Previous work

The Goldreich-Levin (GL) algorithm [18] is considered to be the first algorithm that finds significant Fourier coefficients. The algorithm approximates noisy inner-product functions over $\mathbb{Z}_2^n$, as already mentioned above. An application is a *hardcore function* (known as the GL hardcore bit) for every one-way function. There are two formulations of the GL algorithm. One formulation is due to Rackoff and is based on ideas that were used in earlier work on hardcore bits [4]. The other formulation uses the language of Fourier analysis and was developed in the work of Kushilevitz and Mansour (KM) [25], who extended the ideas to give an algorithm that approximates any real-valued function over $\mathbb{Z}_2^n$. Mansour [32] gave a very similar algorithm for complex-valued functions over $\mathbb{Z}_{2^n}$. All these works rely heavily on the fact that the domain is a group of order 2^n (though this point is not made explicitly in their papers).

For functions on $\mathbb{Z}_N$, Bleichenbacher [8] developed an algorithm very similar in nature to GL and showed how to approximate functions that are a noisy product in $\mathbb{Z}_N^*$, i.e. $f_s(x) = sx + e$. Using this algorithm, Bleichenbacher gave an attack on DSA signatures. Bleichenbacher's work has been used recently to give an attack on nonce leaks in ECDSA [13] and to show some nice results on decomposition techniques in elliptic curves [5].

Akavia, Goldwasser and Safra (AGS) [3] gave a complete algorithm for all complex-valued functions over $\mathbb{Z}_N$, from which it was naturally generalized to all finite abelian groups. Their algorithm can be seen as a generalization of Bleichenbacher's with ideas from KM and Mansour. All these algorithms require the ability to access the function on any requested value, that is, to ask $f(x)$ for any x. We overview them in Section 3.

Applications

Akavia, Goldwasser and Safra [3] showed that a number of bit security results (for RSA, Rabin, and discrete logs) can be re-proved using these tools. A classic result of this type, from Alexi, Chor, Goldreich and Schnorr [4] (ACGS), is that if one has an oracle that on input $x^e \pmod N$ (where (N, e) is an RSA public key) returns the least significant bit of x with probability noticeably better than $\frac{1}{2}$, then one can compute e-th roots modulo N. Håstad and Näslund [23] generalized this result for an oracle that returns any single bit of x (see also [19, Section 4.1]), but their method is very complex and requires complicated and adaptive manipulations of the bits. On the other hand, the algorithm given by AGS, which applies to functions with significant Fourier coefficients, is much clearer and is not adaptive. Similar to Håstad and Näslund, Morillo and Ràfols [37] extended the AGS results to all single bit functions, by showing they have a significant Fourier coefficient (in particular, one can obtain the ACGS result for any bit). Subsequently, a number of papers [14, 15, 16, 48] have proved (or re-proved) various results on bit security in the context of Diffie-Hellman keys on elliptic curves and finite fields $\mathbb{F}_{p^n}$ with $n > 1$, but usually in a model that allows changing the curve or field representation.

We emphasize that the requirement of chosen inputs for the functions restricts these applications. Indeed, the question of main interest, whether single bits of Diffie-Hellman secrets are hardcore, is still open.

The SFT algorithm has also been used to show *search-to-decision* reductions for the *learning with errors* and *learning with rounding* problems [35, 9]. We elaborate on these applications in Section 5.

Paper organisation and contributions

Section 2 summarises the basic definitions. Section 3 presents the key ideas behind the SFT algorithm, and deals with some related issues. Specifically, with few examples we explain why being able to choose the inputs to the functions is essential and why one does not expect to have a similar tool when the inputs to the functions are chosen at random; We also analyze the case of working with unreliable oracles.

In Section 4 we outline our recent work on applying *modulus switching* to this subject (namely to re-cast a function on $\mathbb{Z}_p$ to a function on $\mathbb{Z}_{2^n}$ for the nearest power of 2 to p). These ideas are very similar to the approach taken in Shor's algorithm [41]. In our opinion this provides a simpler approach to the results of Akavia-Goldwasser-Safra and Morillo-Ràfols.

Section 5 surveys bit security applications using the language of the *hidden number problem*: given

f and access to $f_s := f \circ \varphi_s$ find the value s. In Section 6 we explain a fundamental limitation to this approach: we prove that one can only solve the (chosen multiplier) hidden number problem with these tools when the function φ_s is linear or affine. Therefore, these tools cannot be directly used to address the *elliptic curve hidden number problem* or the *modular inversion hidden number problem*. Our work therefore answers a question in [31].

2. PRELIMINARIES

2.1 Fourier analysis on finite groups

We review basic background on Fourier analysis on discrete domains. Proofs and further details can be found in Terras [46].

Let $(R, +, \cdot)$ be a finite ring and denote by $G := (R, +)$ the corresponding additive abelian group. We are interested in the set of functions $L^2(R) := \{f : R \to \mathbb{C}\}$. The set $L^2(R)$ is a vector space over $\mathbb{C}$ of dimension $|G| = |R|$, with the usual pointwise addition and scalar multiplication of functions. Convolution of two functions $f, g \in L^2(R)$ is defined by $(f * g)(x) = \frac{1}{|R|}\sum_{y\in R} f(x-y)g(y)$. The *expectation* of a function $f \in L^2(R)$ is defined to be $\mathbb{E}\,[f] = \frac{1}{|R|}\sum_{x\in R} f(x)$. The space $L^2(R)$ is equipped with an inner product $\langle f, g\rangle := \mathbb{E}\left[f(x)\overline{g(x)}\right] = \frac{1}{|R|}\sum_{x\in R} f(x)\overline{g(x)}$, where $\overline{z}$ denotes the complex conjugate of $z \in \mathbb{C}$. The inner product induces a norm $\|f\|_2 = \sqrt{\langle f, f\rangle}$. We also define $\|f\|_\infty = \max_{x\in R}|f(x)|$.

One basis for this vector space is the set of *Kronecker delta functions* $\{\delta_i\}_{i\in R}$ $\big(\delta_i(j) = 1$ if $j = i$, otherwise $\delta_i(j) = 0\big)$. This is an orthogonal basis with respect to the inner product. However, this basis is not as useful as the Fourier basis, as we will explain later in this section.

A *character* of a group $(G, +)$ is a group homomorphism taking values in the non-zero complex numbers, namely $\chi : G \to \mathbb{C}^*$ such that $\chi(x+y) = \chi(x)\chi(y)$. Since $\chi(x)^{|G|} = \chi(|G|\,x) = \chi(0_G) = 1$, we see that the characters take values in the complex $|G|$-th roots of unity. The set of characters of G forms a group (with respect to point-wise multiplication) which is often denoted $\widehat{G}$, and is isomorphic to G.

In general, we fix a choice of isomorphism $G \to \widehat{G}$ and denote it by $\alpha \mapsto \chi_\alpha$. In particular, for $G = \mathbb{Z}_N$ the characters are defined by $\chi_\alpha(x) := \mathrm{e}^{\frac{2\pi i}{N}\alpha x}$ where $\alpha \in G$. For $G = \mathbb{Z}_{N_1} \times \ldots \times \mathbb{Z}_{N_m}$, let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)$ and $\boldsymbol{x} = (x_1, \ldots, x_m)$; the character $\chi_{\boldsymbol{\alpha}}$ is given by $\chi_{\boldsymbol{\alpha}}(\boldsymbol{x}) := \chi_{\alpha_1}(x_1) \cdot \ldots \cdot \chi_{\alpha_m}(x_m) = \mathrm{e}^{\frac{2\pi i}{N_1}\alpha_1 x_1} \cdot \ldots \cdot \mathrm{e}^{\frac{2\pi i}{N_m}\alpha_m x_m}$ and the map $\boldsymbol{\alpha} \mapsto \chi_{\boldsymbol{\alpha}}$ from G to $\widehat{G}$ is an isomorphism. We sometimes write $\omega_N := \mathrm{e}^{\frac{2\pi i}{N}}$ so that $\chi_\alpha(x) = \omega_N^{\alpha x}$.

The following relations are standard and can be used to show that the characters are orthonormal

$$\sum_{x\in G}\chi(x) = \begin{cases} |G| & \text{if } \chi \text{ is the identity in } \widehat{G}, \\ 0 & \text{otherwise,} \end{cases} \qquad \sum_{\chi\in\widehat{G}}\chi(x) = \begin{cases} |G| & \text{if } x = 0, \\ 0 & \text{otherwise.} \end{cases}$$

If $G = \mathbb{Z}_{N_1} \times ... \times \mathbb{Z}_{N_m}$ then for any $H \leq G$ we define the orthogonal set

$$H^{\perp} := \{a \in G \mid \chi_a(h) = 1 \text{ for all } h \in H\} \ . \tag{1}$$

This set is fundamental for the understanding of the SFT algorithm and appears frequently in Section 3.2. Using the relations above it can be shown that

$$\sum_{h \in H} \chi_h(x) = \begin{cases} |H|, & \text{if } x \in H^{\perp}, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The *Fourier basis* for $L^2(R)$ is the set $\widehat{G}$ consisting of all the characters χ. It is an orthonormal basis. Therefore, we can represent each function $f : R \to \mathbb{C}$ uniquely as a linear combination $f(x) = \sum_{\alpha \in G} \widehat{f}(\alpha)\chi_\alpha(x)$ of the characters χ_α. The function $\widehat{f} : G \to \mathbb{C}$ given by $\widehat{f}(\alpha) = \langle f, \chi_\alpha \rangle$ is called the *discrete Fourier transform*. The map $f \mapsto \widehat{f}(\alpha)$ is $\mathbb{C}$-linear. Notice that a single Fourier coefficient encapsulates information about the function on the whole domain, unlike the representation in terms of Kronecker delta functions where one coefficient only holds information about the function at a single point.

Parseval's identity is the relationship

$$\|f\|_2^2 = \frac{1}{|G|} \sum_{x \in G} |f(x)|^2 = \langle f, f \rangle = \sum_{\alpha \in G} |\widehat{f}(\alpha)|^2 = |G| \cdot \|\widehat{f}\|_2^2$$

between the norms of f and $\widehat{f}$.

Adopting signal processing terminology, when we work with the values $f(x)$ for $x \in G$ we say that x is in the *time domain*. When we use the values $\widehat{f}(\alpha)$ we say $\alpha \in G$ is in the *frequency domain*. There does not seem to be a rigorous formulation of this terminology and we do not use it much, but the reader will find it very common in the engineering literature. We signal to the reader whether we are working in the time domain or frequency domain by using Latin letters x, y for elements in the former (elements of G), and Greek letters α, β for the latter (corresponding to elements of $\widehat{G}$, e.g. χ_α).

Let $f : R \to \mathbb{C}$. Basic properties of the Fourier transform include the following (note that the basis of Kronecker delta functions does not satisfy these properties, which is one reason why it is less useful than the Fourier basis):

- (time) scaling: if $g(x) := f(cx)$ for $c \in R^*$, then $\widehat{g}(\alpha) = \widehat{f}(c^{-1}\alpha)$;
- (time) shifting: if $g(x) := f(c + x)$ for $c \in R$, then $\widehat{g}(\alpha) = \widehat{f}(\alpha)\chi_\alpha(c)$;
- (frequency) shifting: if $g(x) := f(x)\chi_c(x)$ for $c \in R$, then $\widehat{g}(\alpha) = \widehat{f}(\alpha - c)$;
- convolution-multiplication duality: $\widehat{f * g}(\alpha) = \widehat{f}(\alpha)\widehat{g}(\alpha)$.

We now recall some definitions from [3, 14, 37]. The same definitions can be made for functions over rings R where G is their additive group.

Definition 1 (Restriction)**.** Given a function $f : G \to \mathbb{C}$ and a set of characters Γ, the *restriction of f to Γ* is the function $f|_\Gamma : G \to \mathbb{C}$ defined by $f|_\Gamma := \sum_{\chi_\alpha \in \Gamma} \widehat{f}(\alpha)\chi_\alpha$.

Definition 2 (Concentration)**.** Let $\mathscr{J} = \{f_i : G_i \to \mathbb{C}\}_{i\in\mathbb{N}}$ be a family of functions. Let $\epsilon > 0$ be a real number. Then $\mathscr{J}$ is *Fourier ϵ-concentrated* if there exist a polynomial $P \in \mathbb{Z}[s,t]$ and sets of characters $\Gamma_i \subseteq \widehat{G}_i$ such that $|\Gamma_i| \leq P\Big(\frac{1}{\epsilon}, \log|G_i|\Big)$ and $\|f_i - f_i|_{\Gamma_i}\|_2^2 \leq \epsilon$ for all $i \in \mathbb{N}$. We say that $\mathscr{J}$ is *concentrated* if $\mathscr{J}$ is ϵ-concentrated for every $\epsilon > 0$.

Most applications are concerned with a single function that implicitly defines the entire family. In this case we informally say that the function, instead of the family, is concentrated. Examples of concentrated functions, and of this terminology, are given in Example 4.

Definition 3 (Heavy coefficient)**.** For a function $f : G \to \mathbb{C}$ and a threshold $\tau > 0$, we say that a coefficient $\widehat{f}(\alpha)$ (corresponding to the character χ_α) is *τ-heavy* if $|\widehat{f}(\alpha)|^2 > \tau$.

The phrases *significant coefficient* and *heavy coefficient* are often used interchangeably to mean any coefficient $\widehat{f}(\alpha)$ which is large relative to the norm of the function, but without reference to any specific value of τ. By Parseval's identity it is evident that a function cannot have many significant coefficients. In this paper our convention is to use "heavy" in a formal sense and "significant" in an informal sense.

The relationship between concentrated functions and functions with significant coefficients is subtle. If a function has a τ-heavy coefficient, then is it $(1-\tau)$-concentrated (with $|\Gamma| = 1$). But such a function is not necessarily ϵ-concentrated for all ϵ. The literature has tended to focus on concentrated functions, but for many of the bit security applications it is sufficient that the function has one or more significant coefficients. The distinction is important since it is harder to prove that a function is concentrated than to prove it has a significant coefficient.

Example 4. *Here are some examples of functions with significant coefficients, most of which are concentrated:*

- *A single character is concentrated; that is, the family $\mathscr{J} = \{\chi_\alpha : \mathbb{Z}_n \to \mathbb{C}\}_{n>\alpha}$ for some $\alpha \in \mathbb{N}$ is concentrated. The case $\alpha = 0$ corresponds to constant functions, which are concentrated but will be un-interesting in our applications.*

- *For the least-significant-bit function LSB(x) on $\mathbb{Z}_{2^n}$, which gives the parity of x, the functions $f : \mathbb{Z}_{2^n} \to \mathbb{C}$ given by $f(x) := (-1)^{LSB(x)}$ are concentrated. Indeed, these functions correspond to the characters $f(x) = (-1)^x = \omega_{2^n}^{2^{n-1}x} = \chi_{2^{n-1}}(x)$.*

- *The functions half* $: \mathbb{Z}_N \to \{-1,1\}$, *for which half*$(x) = 1$ *if* $0 \le x < \frac{N}{2}$ *and half*$(x) = -1$ *otherwise, are concentrated; one has* $\widehat{half}(\alpha) = \frac{1}{N}[\sum_{0\le x<\frac{N}{2}} \overline{\chi_\alpha}(x) - \sum_{\frac{N}{2}\le x<N} \overline{\chi_\alpha}(x)]$. *Elementary arguments (see Claim 10 below) show that*

$$\left|\frac{1}{N}\sum_{0\le x<\frac{N}{2}} \overline{\chi_\alpha}(x)\right| = \left|\frac{1}{N}\sum_{0\le x<\frac{N}{2}} \omega_N^{-\alpha x}\right| < \frac{1}{|[\alpha]_N|}$$

  where $[\alpha]_N$ *denotes the unique integer in* $(-N/2, N/2]$ *that is congruent to* α *modulo* N. *Similarly* $\left|\frac{1}{N}\sum_{\frac{N}{2}\le x<N} \overline{\chi_\alpha}(x)\right| < \frac{1}{|[\alpha]_N|}$. *These results can be used to show that half is concentrated on a set of characters* α *with small* $|[\alpha]_N|$. *Similar arguments hold for the most-significant-bit function* $f(x) := (-1)^{MSB(x)}$, *thus it is also concentrated.*

- *For primes* p, *the functions* $f : \mathbb{Z}_p \to \mathbb{C}$ *given by* $f(x) := (-1)^{LSB(x)}$ *are concentrated. This follows from* $f(x) = half(2^{-1}x)$ *and the scaling property.*

- *The function* $LPN_s : \{0,1\}^n \to \{0,1\}$, *given by* $LPN_s(x) = (-1)^{\langle x,s\rangle + e(x)}$ *for* e *which is mostly* 0 *(and otherwise* 1*), has a significant coefficient and therefore is* ϵ*-concentrated (for some large* ϵ*). Let* I *be the set for which* $e(x) = 1$, *then* $\widehat{LPN_s}(s) = \frac{1}{2^n}\sum_{x\notin I} 1 + \frac{1}{2^n}\sum_{x\in I}(-1) = 1 - \frac{2|I|}{2^n}$. *Since the size* $|I|$ *is relatively small, the coefficient* $\widehat{LPN_s}(s)$ *is large, that is, the function* LPN_s *"behaves" like the character* χ_s *in* $\{0,1\}^n$. *If* $|I|$ *is very small, for example* $|I| = poly(\log|G|)$, *then* LPN_s *is also concentrated. Moreover, one can show that* $|\widehat{LPN_s}(v)| \le \frac{|I|}{2^n}$, *and on average is expected to be proportional to* $\sqrt{2|I|/2^n(2^n-1)} \approx \sqrt{2|I|}/2^n$.

- *'Noisy characters' given by* $f(x) := \omega_p^{\alpha x + e(x)}$ *for some suitable random functions* e *have a significant coefficient* $\widehat{f}(\alpha)$ *as we show in Section 6.1. An example of such a noisy character is the function* $LWE_s : \mathbb{Z}_p^n \to \mathbb{Z}_p$, *given by* $LWE_s(x) = \omega_p^{\langle x,s\rangle + e(x)}$ *for* $e(x)$ *drawn from a Gaussian distribution.*

Another example of concentrated functions are the i-th bit functions, see Section 4.1 for details.

2.2 Learning model

Let $f : R \to \mathbb{C}$ be a function for which one wants to learn its significant coefficients. The learner gets *access* to samples of the form $(x, f(x))$. In the *random access* model the learner receives polynomially many samples for inputs $x \in R$ drawn independently and uniformly at random. As opposed to this model, in the *query access* model the learner can query the function on any chosen input $x \in R$ to receive the corresponding sample.

An algorithm *learns* a function f if it outputs a set containing all its significant Fourier coefficients. Formally, given a function f and $\epsilon, \delta > 0$, the algorithm outputs a set Γ, such that $\|f - f|_\Gamma\|_2^2 \le \epsilon$ with probability at least $1 - \delta$.

The main result of this subject (see Theorem 6 below) is that there is a randomised, polynomial-time algorithm to compute a sparse approximation $f|_\Gamma$ to a concentrated function in the query access model. In other words, concentrated functions are learnable in polynomial time.

2.3 Probability

The Chernoff bound gives an upper bound on the probability that a sum of independent random variables deviates from its expected value. One can therefore derive a lower bound for the number of samples needed to estimate the sum of independent random variables, with any required probability and error term. For a random variable X on a set $A \subseteq \mathbb{C}$ we denote by $\mathbb{E}_{x \in A} X(x)$ the expected value $\sum_{x \in A} X(x) \Pr(x)$.

Theorem 5 (Chernoff)**.** *Let A be a set of complex numbers such that $|x| \leq M$ for all $x \in A$. Let $x_i \in A$ be chosen independently and uniformly at randomly from A. Then*

$$Pr\left[\left|\mathop{\mathbb{E}}_{x \in A}[x] - \frac{1}{m}\sum_{i=1}^{m} x_i\right| > \lambda\right] \leq 2\mathrm{e}^{-\lambda^2 m/2M^2} \quad .$$

2.4 Table of notations

We summarize the main notation and definitions in the following table.

Notation/Definition	Meaning
χ	A character of G.
$H^\perp$	The orthogonal set $\{\alpha \in G \mid \chi_\alpha(h) = 1 \text{ for all } h \in H\}$.
$\widehat{f}$	The Fourier transform of f.
Scaling property	$\widehat{g}(\alpha) = \widehat{f}(c^{-1}\alpha)$ for $g(x) := f(cx)$ and $c \in R^*$.
$\tau-$heavy coefficient	A coefficient satisfying $\lvert\widehat{f}(\alpha)\rvert^2 > \tau$.
Significant coefficient	A $\tau-$heavy coefficient, for some $\tau^{-1} = poly(\log \lvert G\rvert, \lVert f\rVert_\infty)$.
Query access	The ability to ask for $f(x)$ for any input x.

3. The SFT Algorithm

This section gives a high-level presentation of the significant Fourier transform (SFT) algorithm in the query access model. A precise formulation of the algorithm is given in Theorem 6. The section starts with simple algorithms for specific functions with only one significant coefficient. This is in order to show that the general algorithm is a natural extension of these simple algorithms. We then present the algorithm for domains of size 2^n (which is also applicable for other domains with towers of subgroup of small index), and finally show which modifications should be done for domains of prime order.

Theorem 6 ([1, SFT algorithm][3, Theorem 5]). *Let G be an abelian group represented by a set of generators of known orders. There is a learning algorithm that given query access to a function $f : G \to \mathbb{C}$, a threshold $\tau > 0$ and $\delta > 0$, finds all the τ-heavy Fourier coefficients of f with probability at least $1 - \delta$. The algorithm runs in polynomial time in $\log\left(|G|\right)$, $\|f\|_\infty$, $\frac{1}{\tau}$ and $\log\left(\frac{1}{\delta}\right)$.*

3.1 History and special cases

Key ideas behind the SFT algorithm first arose in other settings, and the aim of this section is to put some of this early work in context.

3.1.1 Goldreich-Levin

Consider a noisy inner product function. That is, $f_s : \{0,1\}^n \to \{0,1\}$ given by $f_s(x) = \langle x, s\rangle + \delta(x)$ (addition takes place mod 2) where $\delta(x) = 1$ with some small probability (noticeably smaller than $\frac{1}{2}$) and otherwise $\delta(x) = 0$. This is the same function as in the learning parity with noise (LPN) problem. If $\delta(x) = 0$ for all x (or for a negligible set of inputs) then reconstructing f_s is an easy linear algebra problem. The task is to learn s given samples $f_s(x_i)$. In the simplest setting of Goldreich and Levin [18] there is a single τ-heavy character $\chi_s(x) = (-1)^{\langle x,s\rangle}$ for $\tau > 1/2$.

If one can choose the queries for f_s then an elementary approach is to query on the unit vectors $e_1 := (1, 0, \ldots, 0), \ldots, e_n := (0, \ldots, 0, 1)$ to learn s bit-by-bit. However, since the query on e_i may return the incorrect answer $\langle e_i, s\rangle + 1$, one would like to generate a small set of independent values of the form $\langle e_i, s\rangle + \delta$, and determine s_i by majority rule. This can simply be achieved by querying on 'consecutive' values x and $x + e_i$ to get the results $\langle x, s\rangle + \delta(x)$ and $\langle x, s\rangle + \langle e_i, s\rangle + \delta(x + e_i)$. If both answers are not noisy (or if both are noisy) then by subtracting one from the other we get $\langle e_i, s\rangle$, which is the i-th coordinate of s. (For the interested reader: if the noise rate approaches $\frac{1}{2}$, then there will not be a unique solution; Rackoff suggested to use a trick due to Alexi et al. [4] to deal with this case.)

The original Goldreich-Levin paper [18] does not give a clear description of the learning algorithm. A description in the language of Fourier analysis was given in [25] by Kushilevitz and Mansour.

Note that the problem of learning s from noisy inner products $\langle x, s\rangle$ can be interpreted as decoding a binary linear code. Choosing the queries x can be interpreted as designing a generator matrix for the code. The SFT algorithm can therefore be viewed as a decoding algorithm. In the situation where the error rate is very high and there is a not a uniquely determined solution then the SFT algorithm can be viewed as a list-decoding algorithm. More on the relation of these tools to decoding linear codes can be found in the recent work [24], where an 'extended KM' algorithm is presented.

3.1.2 Bleichenbacher

Bleichenbacher [8] seems to have been the first to consider these problems in the case of functions on $\mathbb{Z}_N$ where N is not a power of 2. He considers a 'noisy product' function $f_s : \mathbb{Z}_N \to \mathbb{Z}_N$ given by $f_s(x) = sx + \delta(x)$ where with high probability $|\delta(x)| < \frac{N}{2^\lambda}$ for some real number λ. If $\delta(x)$ is very small then finding s and reconstructing f_s is easy.

In Bleichenbacher's original setting one cannot choose the queries, so he gives a method (not efficient for large domains) to obtain queries that lie in short intervals, and then gives a method to solve the problem. We explain the latter method. If one can obtain any query, then this problem can be solved by successively multiplying by 2 and reading the bits. Since some samples may be erroneous, a majority rule idea, based on the approach taken in GL above, is used.

The main idea to solve this problem comes from the fact that if $s < \frac{N}{2^\eta}$, for some $\eta \geq 0$, then $sy < N$ for every $0 \leq y \leq 2^\eta$. In other words, the product sy does not 'wrap-around' modulo N, and this can be used to determine the upper bits of s: given y and $f_s(y) = sy + \delta(y)$, take $\lfloor f_s(y)/y \rceil = \lfloor s + \delta(y)/y \rceil$; assuming there is no wrap-around over N in $f_s(y)$, we get some upper bits of s. Now suppose one knows $\mathrm{MSB}_\eta(s)$, the η most-significant bits of s, then by subtracting it from s we have $s - \mathrm{MSB}_\eta(s) < \frac{N}{2^\eta}$. One can then query on $2^{\eta-1} \leq y \leq 2^\eta$ to determine the next bit.

Notice that this approach requires having multipliers drawn from some interval $\{0, 1, \ldots, 2^i - 1\}$ (specifically small multipliers in the first stages, which are the 'hardest' to get). As some samples are incorrect, we need to generate independent sets of this form. Similar to the approach in GL, this is done by fixing some z and querying on $z + r$ for r chosen uniformly in $\{0, 1, \ldots, 2^\eta - 1\}$, then subtracting.

This is a simplification of Bleichenbacher ideas, which actually involve Fourier analysis. For the full details we refer to Bleichenbacher [8]. This method does not seem to have been used for cryptographic applications until the recent work [13, 5].

3.1.3 Following work

The early work did not explicitly mention Fourier coefficients, but it was realised that one can re-phrase the problems as finding significant Fourier coefficients of these functions. The Goldreich-Levin case was generalized by Kushilevitz and Mansour [25] to any real-valued function over $\{0, 1\}^n$ and this work was the first to explicitly treat functions with more than one significant Fourier coefficient.

Subsequently, Mansour [32] gave an algorithm for functions $f : \mathbb{Z}_{2^n} \to \mathbb{C}$. Unlike other works, Mansour finds the significant coefficients from the least-significant bit to the most-significant bit (a link between these works is explained in Remark 8 below). The approach of Mansour was extended, thereby giving a generalisation of Bleichenbacher's result, by Akavia, Goldwasser and Safra [3].

Notice that combining the KM and AGS ideas gives an algorithm for all groups $\mathbb{Z}_{N_1} \times \cdots \times \mathbb{Z}_{N_r}$, since one can easily collapse from the latter to $\mathbb{Z}_{N_j}$ (by choosing appropriate queries). Therefore, the case of most interest is $G = \mathbb{Z}_p$ as we present below. As further evidence for the unity of all these ideas we remark that the KM and AGS algorithms query on exactly the same set of queries as GL and Bleichenbacher (and subsequently reveal the significant coefficients bit-by-bit from MSB to the LSB).

3.2 The SFT algorithm

Let $f : G \to \mathbb{C}$. Given a threshold $\tau \in \mathbb{R}$, the algorithm outputs all τ-heavy Fourier coefficients of f (and potentially some other coefficients, which are close in size to τ) with overwhelming probability.

We first give a high level view of how the algorithm works. The method is a form of binary search: the algorithm divides the set of Fourier coefficients into two (disjoint) sets, and check each set separately to determine whether it potentially contains a τ-heavy coefficient. To do this the algorithm defines two new functions, one for each set of coefficients. A clever use of Parseval's identity allows the algorithm to check the size of all coefficients simultaneously given the norm of the functions. Hence, the problem is to determine the norms of the two new functions, which requires a method to compute their values. The structure of the sets is important: for some sets we have useful formulas for the functions. Instead of precisely calculating these values, it is sufficient to have approximations of the outputs of the functions and to approximate the norm of each function. The Chernoff bound is then used to bound the error term in the approximations.

Schematically, the algorithm operates as follows, where we initially take $D = G$:

- Partition $D = A \uplus B$, and define $f_A(x) := f|_A = \sum_{\alpha \in A} \widehat{f}(\alpha)\chi_\alpha(x)$ and $f_B(x) := f|_B = \sum_{\beta \in B} \widehat{f}(\beta)\chi_\beta(x)$.
- Approximate the values $f_A(x_i)$ and $f_B(y_j)$ for polynomially many samples x_i, y_j, chosen uniformly at random. This is done using the fundamental relation in (3) below.
- Using the values from the previous stage, approximate the norms $||f_A||_2^2$ and $||f_B||_2^2$. See (5).
- Using Parseval's identity $||f_A||_2^2 = \sum_{\alpha \in A} |\widehat{f}(\alpha)|^2$, if the approximation of the norm is smaller than[1] $\frac{\tau}{2}$ then with overwhelming probability f does not have a τ-heavy coefficient in A. Hence, dismiss A. Similarly for f_B.
- Run the algorithm recursively on the remaining sets and stops when it reaches singletons.

As already explained, Parseval's identity shows that a function can only have polynomially many significant coefficients. Hence, by setting a threshold τ such that $\tau^{-1} = poly(\log |G|, \|f\|_\infty)$, the number

[1] A lower threshold $\frac{\tau}{2}$ is needed since the algorithm only approximates the norm. As a consequence, the final list may contain coefficients that are $\frac{\tau}{2}$-heavy but not τ-heavy.

of sets involved in the process (therefore the number of iterations) is polynomial (see [25, Lemma 3.4] or [33, Lemma 4.8])[2].

Remark 7. *We emphasize that the algorithm can work with any function f and with any threshold τ. Specifically, if f does not have any τ-heavy coefficients, then the algorithm will output an empty list. However, the running time is polynomial in $1/\tau$ so the algorithm will not be efficient if the threshold is chosen to be too low. Indeed, if τ is small then the algorithm will insist on using many samples to get sufficiently close approximations to the norms.*

To illustrate these points, consider a function that has a coefficient $\widehat{f}(v)$ that is relatively large compared to each of other coefficients but is not significant (for example 10 times larger than each of the rest), i.e. it is not large relative to the norm. Suppose one tries to find this coefficient by setting a very low threshold. The running time of the algorithm on this function would not be polynomial as, at the first stages, the sums of all Fourier coefficients over the sets are roughly the same size. Therefore, the algorithm would have to keep all the sets until they are sufficiently small. This case corresponds to a τ-heavy $\widehat{f}(v)$ for τ^{-1} that is not polynomial in $\log |G|, \|f\|_\infty$. To get that τ^{-1} is polynomial in $\log |G|, \|f\|_\infty$, the size of $\widehat{f}(v)$ should be comparable with the norm of the function and not just larger than all the rest.

3.2.1 Domains of size 2^n

We now sketch an algorithm that unifies the KM and Mansour algorithms. Our presentation is more group-theoretic than the original works. We refer to [25] and [33] for the proofs.

Let $f : G \to \mathbb{C}$ and $\tau \in \mathbb{R}$. At each iteration the algorithm takes a set D (starting with $D = G$) and proceeds as follows.

Partial functions. Partition $D = A \cup B$ into two sets that are defined below. Define the function $f_A : G \to \mathbb{C}$ by $f_A(x) = \sum_{\alpha \in A} \widehat{f}(\alpha)\chi_\alpha(x)$. If f has a τ-heavy coefficient α and $\alpha \in A$, then f_A has a τ-heavy coefficient. All arguments hold similarly for the set B.

Estimating f_A. We need a method to estimate values of the function f_A using values of the original function f. We define a *filter function* $h_A : G \to \mathbb{C}$ by $h(x) = \sum_{\alpha \in A} \chi_\alpha(x)$, and then use the property $\widehat{f * h_A} = \widehat{f} \cdot \widehat{h_A}$. Since

$$\widehat{h_A}(\alpha) = \begin{cases} 1 & \alpha \in A, \\ 0 & \text{otherwise}, \end{cases}$$

we have

$$\widehat{f * h_A}(\alpha) = \begin{cases} \widehat{f}(\alpha) & \alpha \in A, \\ 0 & \text{otherwise}. \end{cases}$$

[2] The notion of heavy coefficient in [25, 33] is slightly different from ours. There, a coefficient is τ-heavy if $|\widehat{f}(\alpha)| > \tau$.

In other words,

$$f * h_A = f_A. \tag{3}$$

Convolution is not a task we have an efficient method to calculate in general, let alone efficiently calculating $h_A(x) = \sum_{\alpha \in A} \chi_\alpha(x)$. Therefore, the structure of the sets is important and plays a key role in the ability to apply the algorithm. Notice that if A is an arithmetic progression, then $\sum_{\alpha \in A} \chi_\alpha(x) = \sum_j \chi_{qj+r}(x) = \chi_r(x) \sum_j \chi_q(jx)$, and so it can be evaluated by the formula for geometric series. More generally, assume $D \leq G$ is a subgroup and let $H \leq D$ be a subgroup (of index 2). We take A to be a coset $A = z + H$ for some $z \in G$ (then B is taken to be the other coset). Then,

$$h_{z+H}(x) = \sum_{h \in H} \chi_{z+h}(x) = \sum_{h \in H} \chi_z(x)\chi_h(x) = \chi_z(x) \sum_{h \in H} \chi_h(x),$$

and the latter is zero unless $x \in H^\perp$ ($H^\perp$ is defined in (1) above). Thus the function h_A is given by

$$h_A(x) = h_{z+H}(x) = \begin{cases} \chi_z(x) \cdot |H|, & \text{if } x \in H^\perp, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We therefore get, since $|H||H^\perp| = |G|$,

$$\begin{aligned} f_A(x) = f * h_A(x) &= \mathop{\mathbb{E}}_{y \in G} \left[f(x-y) h_A(y) \right] = \frac{1}{|G|} \sum_{y \in G} f(x-y) h_A(y) \\ &= \frac{1}{|G|} |H| \sum_{y \in H^\perp} f(x-y)\chi_z(y) = \mathop{\mathbb{E}}_{y \in H^\perp} \left[f(x-y)\chi_z(y) \right] \ . \end{aligned}$$

The last term is an expectation over values of size at most $\|f\|_\infty$, and so the Chernoff bound guarantees that polynomial (in $\log(|G|)$) many samples (chosen uniformly at random in $H^\perp$) are sufficient to approximate it with an error term of size at most $\frac{\|f\|_\infty}{poly(\log(|G|))}$ with overwhelming probability.
We give concrete examples of this stage in Section 3.2.2 below.

Estimating $\|f_A\|_2$. We can now write $\|f_A\|^2$ as

$$\|f_A\|_2^2 = \mathop{\mathbb{E}}_{x \in G} \left| (f * h_A)(x) \right|^2 = \mathop{\mathbb{E}}_{x \in G} \left| \mathop{\mathbb{E}}_{y \in G} \left[f(x-y) h_A(y) \right] \right|^2 = \mathop{\mathbb{E}}_{x \in G} \left| \mathop{\mathbb{E}}_{y \in H^\perp} \left[f(x-y)\chi_z(y) \right] \right|^2 \ .$$

Again, an approximation of the norm is sufficient (a consequence of the approximation is that we have to lower the threshold τ a little bit).

We can therefore approximate $\|f_A\|_2^2$ by choosing m_1, m_2 sufficiently large (given by the Chernoff bound), randomly choosing[3] $x_i \in G$ where $1 \leq i \leq m_1$, randomly choosing $y_{ij} \in H^\perp$ for each i where $1 \leq j \leq m_2$ and calculating

$$\frac{1}{m_1} \sum_{i=1}^{m_1} \left| \frac{1}{m_2} \sum_{j=1}^{m_2} f(x_i - y_{ij})\chi_z(y_{ij}) \right|^2 \approx \|f_A\|_2^2 = \sum_{\alpha \in A} |\widehat{f}(\alpha)|^2 \ . \tag{5}$$

[3]Note that as in [25, 32] one can define the function f_A over H (and not G), and therefore choose the values x_i from H.

One then checks if this value is smaller than $\tau/2$. If so then with overwhelming probability there is no $\alpha \in A$ such that $\widehat{f}(\alpha)$ is τ-heavy, and so the set A can be dismissed. Notice that if this value is greater than $\tau/2$ it does not necessarily mean that A contains a significant coefficient. In this case the algorithm sets $D = A$ and repeats until all sets are singletons. As mentioned above, as long as the threshold τ satisfies $\tau^{-1} = poly(\log |G|, \|f\|_\infty)$, it is guaranteed that the number of sets the algorithm keeps throughout the process is polynomial in $\log(|G|)$.

We give the pseudocode of the algorithm. At start, set $k = n$, so $H_k = G$.

Algorithm 1: MainProcedure

Input: A coset $z + H_k$.
if $|H_k| = 1$ **then**
  if Est $\widehat{f}(z) \geq \tau/2$ **then**
    return $\{z\}$
  else
    return $\emptyset$
else
  Let W be a set of coset representatives for H_{k-1} in H_k
  Let $W' = \{w \in W \mid \text{EstNormSq}(f_{(z+w)+H_{k-1}}) \geq \tau/2\}$
  return $\cup_{w \in W'}$ MainProcedure$((z + w) + H_{k-1})$

Algorithm 2: EstNormSq

Input: $f_{z+H} : G \to \mathbb{C}$.
Choose $x_i \in G$ where $1 \leq i \leq m_1$
For each i, choose $y_{ij} \in H^\perp$ where $1 \leq j \leq m_2$
return $\frac{1}{m_1} \sum_{i=1}^{m_1} \left| \frac{1}{m_2} \sum_{j=1}^{m_2} f(x_i - y_{ij}) \chi_z(y_{ij}) \right|^2$

Algorithm 3: Est$\widehat{f}$

Input: $z \in G$.
Choose $x_i \in G$ where $1 \leq i \leq m_1$
return $\frac{1}{m_1} \sum_{i=1}^{m_1} f(x_i) \chi_z(-x_i)$

3.2.2 Examples

Notice that in (5) above for each x_i one needs the samples $f(x_i - y_{ij})$. This explains the importance of having query access to the function. To illustrate this point, we give some concrete examples.

Kushilevitz and Mansour [25] consider a function $f : \{0, 1\}^n \to \mathbb{R}$. Write $x = x_1 \ldots x_n$. At the first iteration define A to contain all n-bit strings that start with 0 and B to contain all the n-bit strings that start with 1. Then we have

$$h_A(x) = \begin{cases} 2^{n-1}, & \text{if } x = 0\ldots 0 \text{ or } x = 10\ldots 0, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

and indeed

$$\widehat{h_A}(\alpha) = \frac{1}{2^n}\sum_x h_A(x)(-1)^{\langle \alpha, x\rangle} = \frac{1}{2}\Big((-1)^0 + (-1)^{\alpha_1}\Big) = \begin{cases} 1 & \alpha \in A; \\ 0 & \text{otherwise.} \end{cases}$$

One can only evaluate $f * h_A(x)$ if one has the values $f(x)$ and $f(x + e_1)$. This shows that the KM approach requires (in the first iteration) queries on pairs of vectors that differ by a unit vector, exactly as in the elementary approach to the GL theorem as sketched in Section 3.1.1.

Mansour [32] considers a function $f : \mathbb{Z}_{2^n} \to \mathbb{C}$. At the first iteration define A to contain all the even numbers in $\mathbb{Z}_{2^n}$ and B to contain all the odd numbers. Then, we have

$$h_A(x) = \begin{cases} 2^{n-1}, & \text{if } x = 0 \text{ or } x = 2^{n-1}, \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

and indeed

$$\widehat{h_A}(\alpha) = \frac{1}{2^n}\sum_x h_A(x)\omega_{2^n}^{\alpha x} = \frac{1}{2}\left(1 + (-1)^{\alpha}\right) = \begin{cases} 1 & \alpha \in A; \\ 0 & \text{otherwise.} \end{cases}$$

One can only evaluate $f * h_A(x)$ if one has $f(x)$ and $f(x + 2^{n-1})$.

The analysis of this algorithm is useful for the prime case below, and so we present its later stages. In stage l of this algorithm, one defines the subgroup H to contain all multiples of 2^l in $\mathbb{Z}_{2^n}$. Hence the cosets in a partition contain all numbers that agree on their remainder modulo 2^l, and $H^\perp = \{x \in \mathbb{Z}_{2^n} \mid x2^l \equiv 0 \pmod{2^n}\} = \{0, 2^{n-l}, 2\cdot 2^{n-l}, 3\cdot 2^{n-l}, \ldots, (2^l - 1)2^{n-l}\}$. Define $A = A_r = \{x \in \mathbb{Z}_{2^n} \mid x \equiv r \pmod{2^l}\} = H + r$. Then, the filter function h_A satisfies

$$h_A(x) = \begin{cases} \chi_r(x)\cdot 2^{n-l}, & \text{if } x \in H^\perp, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Again, to approximate $f * h_A(x)$, one needs enough samples $f(x_i)$ for $x_i \in H^\perp$.

Remark 8. *Readers familiar with lattice cryptography may be interested to know that one can view the relationship between the KM [25] algorithm on $\{0,1\}^n$ and the Mansour [32] algorithm on $\mathbb{Z}_{2^n}$ as the modulus-dimension tradeoff [28]. We briefly sketch this idea. Let $\boldsymbol{a} = (a_0, \ldots, a_{n-1}) \in \mathbb{Z}_p^n$, $\boldsymbol{s} = (s_0, \ldots, s_{n-1}) \in \{0,1\}^n$, and suppose*

$$b \equiv \boldsymbol{a}\cdot\boldsymbol{s} + e \equiv \sum_{i=0}^{n-1} a_i s_i \ + \ e \pmod{p}.$$

Writing $a = a_0p^{n-1} + a_1p^{n-2} + \cdots + a_{n-2}p + a_{n-1}$ *and* $s = s_0 + s_1p + \cdots + s + n - 1p^{n-1}$ *we have*

$$as \equiv (a_0s_0 + \cdots + a_{n-1}s_{n-1})p^{n-1} + \textit{lower term} \pmod{p^n}$$

and some of its MSBs agree with the MSBs of bp^{n-1}*, when* p *is large.*

As shown in equation (6) above, at the first iteration over $\{0,1\}^n$ *the filter function is nonzero on the inputs* ***0*** *and* ***a*** $= (1, 0, \ldots, 0)$ *in* $\mathbb{Z}_2^n$*. These vectors correspond to the values* $a = 0$ *and* $a = 2^{n-1}$ *in* $\mathbb{Z}_{2^n}$*, which are exactly the values appearing in equation (7). Since the lower terms of* as *are zero, when* $a = 0, p^{n-1}$*, the MSB of* as *and* bp^{n-1} *agree even for* $p = 2$*. In both domains, we use these values to recover* s_0*. The generalization to all inputs* ***a*** *arising in the algorithms is straightforward.*

3.2.3 Domains of prime order

The algorithm presented above is suitable for domains whose order can be factored as a product of small primes (especially for powers of 2, as been shown for $\{0,1\}^n$ in [25] and for $\mathbb{Z}_{2^n}$ in [32]). A case of interest, from the theoretical and practical sides, is domains of (large) prime order. Since each additive group $\mathbb{Z}_N$ can be decomposed into $\mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n}$, and since we have query access (so naively we can query on $\mathbf{x} = (0, \ldots, 0, x, 0, \ldots, 0)$ to 'filter' all other subgroups),[4] being able to find heavy coefficients for functions over $\mathbb{Z}_p$ will allow us to find heavy coefficients for functions over any $\mathbb{Z}_N$.

One cannot apply directly the algorithm presented above as $\mathbb{Z}_p$ does not have any proper subgroups, specifically not those of small index. The importance of the subgroups is in the evaluation of exponential sums (such as equation (2) above), which subsequently allows us to have useful formulas for the filter functions such as equation (4)). We now show that one can still follow the steps in the algorithm above. Natural candidates for the partitioning sets are classes of numbers with the same remainder modulo 2^l (where l represents the level we work at), which is similar to the approach taken over $\mathbb{Z}_{2^n}$ (see Section 3.2.2), or intervals (of similar size) of consecutive numbers.[5] In fact, using the frequency-shifting and scaling properties of the Fourier transform, one can show that these two partitions are equivalent (where there is a correspondence between the size of the intervals and the size of the classes), in the sense that one can transform the coefficients in an interval to coefficients of the same class modulo 2^l and vice versa.

The algorithm over $\mathbb{Z}_p$ works in the same stages as explained in Section 3.2. The main obstacle is to show how to efficiently calculate the function f_A, for some appropriate set A (one should also make sure that one can have a good approximation of $\|f_A\|_2$, but this does not turn out to be an issue). We therefore focus on this stage. The other stages are similar to the algorithm for domains of size 2^n.

Working in the 'frequency domain'. In order to show the difficulty working in a domain of prime size, we start with a naive imitation of the approach taken in the algorithm for domains of size 2^n. Let A

[4] Since deterministic queries are not desirable, additional randomization is used in practice.

[5] Note that both are arithmetic progressions, which allow evaluating h_A.

be an arithmetic progression in $\mathbb{Z}_p$, and define $f_A = \sum_{\alpha\in A} \widehat{f}(\alpha)\chi_\alpha(x)$ and $h_A(x) = \sum_{\alpha\in A} \chi_\alpha(x)$ as above. Then $f_A(x) = f * h_A(x) = \mathbb{E}_{y\in G}\left[f(x-y)h_A(y)\right] = \mathbb{E}_{y\in G}\left[f(x-y)\sum_{\alpha\in A}\chi_\alpha(y)\right]$. Since A is an arithmetic progression, $\sum_{\alpha\in A}\chi_\alpha(x)$ is a geometric progression for which we have a formula. We get that $f_A(x)$ is an expectation over values each of which we can calculate exactly. Moreover, unlike in the algorithm above, the filter function here is nonzero over a very large set, and therefore one can hope that specific queries are not needed in this case. This turns out to be a downside. Indeed, in order to determine $f_A(x)$ in polynomial time, we can only approximate this expectation, but as the values of this geometric progression can be as large as $|A|$, one derives from the Chernoff bound that the number of samples needed to have a good approximation of $f_A(x)$ is roughly $|A|$, which is exponential in $\log(p)$ in the first stages of the algorithm. Hence this approach is not practical.

Working in the 'time domain'. Instead of working in the 'frequency domain', we can work in the 'time domain'. In this case we define A to be a class of numbers with the same remainder mod 2^l. We adapt the filter function in (8) to the $\mathbb{Z}_p$ case. As in Section 3.2.2, let H be the set containing all multiples of 2^l in $\mathbb{Z}_p$. Define $H^\perp := \{0, 2^{-l}, 2\cdot 2^{-l}, \ldots, \lfloor \frac{p}{2^l}\rfloor 2^{-l}\}$. Notice that while $H^\perp$ is not orthogonal to H, it contains all numbers that give small remainder (mod p) when multiplied by 2^l. Let $z \in \mathbb{Z}_p$ such that $z \equiv r \pmod{2^l}$ and define $A = A_r = \{x \in \mathbb{Z}_p \mid x \equiv r \pmod{2^l}\}$ to be the class in $\mathbb{Z}_p$ for which the remainder mod 2^l is r. We define

$$h_A(x) = h_{z+H}(x) = \begin{cases} \frac{p}{2^l}\chi_z(x), & \text{if } x \in H^\perp, \\ 0, & \text{otherwise.} \end{cases}$$

It turns out that this function, which is a simple adaptation of (8) to $\mathbb{Z}_p$, is a 'noisy' version of a 'pure' filter function: the size of the coefficients $|\widehat{h_A}(\alpha)|$ is close to 1 for $\alpha \in A$ and close to 0 for $\alpha \notin A$. Indeed,

$$\widehat{h_A}(\alpha) = \frac{1}{p}\sum_{x\in\mathbb{Z}_p} h_A(x)\overline{\chi_\alpha}(x) = \frac{1}{2^l}\sum_{x\in H^\perp}\chi_{z-\alpha}(x) \ .$$

Write $\alpha = 2^l k + j$, $z = 2^l q + r$ and $x = d2^{-l}$ for $0 \le j, r < 2^l$ and $0 \le d \le \lfloor \frac{p}{2^l}\rfloor$. Then,

$$\widehat{h_A}(\alpha) = \frac{1}{2^l}\sum_{0\le d\le\lfloor\frac{p}{2^l}\rfloor}\chi_{2^l q + r - 2^l k - j}(d2^{-l}) = \frac{1}{2^l}\sum_{0\le d\le\lfloor\frac{p}{2^l}\rfloor}\chi_{q-k}(d)\chi_{r-j}(2^{-l}d) \ .$$

One can show that the last sum is large if and only if $j = r$ as $\chi_{r-j} \equiv 1$ – that is, if and only if $\alpha \in A$ – and so that $|\widehat{h_A}(\alpha)| \approx 1$, and otherwise it is close to 0. More precisely, for $\alpha = z$ we have $|\widehat{h_A}(\alpha)| = 1$ and as k gets further away from q, the size of $\widehat{h_A}(2^l k + r)$ slowly decays (follows from Claim 10 below). The function h_A is said to be "centered around" z.

As mentioned above, using the scaling and frequency-shifting properties one can transform from the set A to an interval I of the same size. That is, define $h_I(x) := h_A(2^{-l}x)$, then $\widehat{h_I}(\alpha) = \widehat{h_A}(2^l\alpha)$.

This is a permutation of the coefficients of h_A. If $A = \{r, 2^l + r, \ldots, t2^l + r\}$, then $I = \{r2^{-l}, r2^{-l} + 1, \ldots, r2^{-l} + t\}$, and the coefficients which were large on A and small outside A are now large over I and small outside it. The approach taken in [3, 1] is to work over intervals. For an interval $[a, b]$ of size $\lfloor \frac{p}{2^l} \rfloor$, for which $c = \lfloor \frac{a+b}{2} \rfloor$ is a middle point, one defines

$$h_{a,b}(x) = \begin{cases} \frac{p}{2^l}\chi_c(x), & \text{if } 0 \leq x < 2^l, \\ 0, & \text{otherwise.} \end{cases}$$

A direct calculation using the definition of $\widehat{h}_{a,b}(\alpha)$ shows that

$$\widehat{h}_{a,b}(\alpha) = \mathop{\mathbb{E}}_{0 \leq x < 2^l} \left[\chi_c(x) \overline{\chi_\alpha(x)} \right] = \mathop{\mathbb{E}}_{0 \leq x < 2^l} \left[\chi_{c-\alpha}(x) \right].$$

Again, one can show that $|\widehat{h}_{a,b}(\alpha)| \approx 1$ if $a \leq \alpha \leq b$ and $|\widehat{h}_{a,b}(\alpha)| \approx 0$ for α outside this interval (see, for example, Claim 10). For further details see [3, 1]. This function is "centered around" c, that is, for $\alpha = c$ we have $|\widehat{h_A}(\alpha)| = 1$ and while α gets further away from c, the size of $\widehat{h_A}(\alpha)$ slowly decays.

Remark 9. *There is a technical issue which we ignore in this paper. As the size of $\widehat{h_A}(\alpha)$ slowly decays while α moving away from c, when α reaches the end of the interval $[a, b]$ the size of $\widehat{h_A}(\alpha)$ is close to the size of $\widehat{h_A}(\beta)$ for β just outside this interval. This imposes some complexities in the filtering process; specifically one should take overlapping intervals, so the sets A, B are not distinct as in the case of domains of size 2^n. Moreover, the choice of the point c (therefore the choice of the interval) also affects the filtering process. We refer to Sections 7.2.3 and 7.2.4 in [3] and to [1, Section 3] for the technical details.*

With this filter function (either h_A or $h_{a,b}$) f_A can be approximated efficiently, as shown in the previous section. The algorithm now proceeds as the algorithm for domains of size 2^n.

3.3 Working with unreliable oracles

It is sometimes desirable to describe access to the function f as querying an oracle. The oracle can be perfect – always provides the correct value $f(x)$ – or imperfect. Working with unreliable oracles is of importance in several applications. This section is dedicated to analyzing these cases.

Sometimes the samples $f(x_i)$ are given by an unreliable oracle O. By this we mean the oracle satisfies $O(x) = f(x)$ only with high probability. One can think of O as a 'noisy version' of f. A common approach to this situation is to generate several independent values, each of which gives the value $f(x)$ with good probability; then, by applying majority rule, one can obtain the correct value $f(x)$ with overwhelming probability. Examples of this approach are presented in Section 3.1.

We show how the language of Fourier analysis gives a very general approach to analyze situations for working with unreliable oracles. The main idea is that if a function f has a significant Fourier coefficient, then its noisy version also has a significant coefficient. Note however that if f is concentrated, then its 'noisy' version is not necessarily concentrated.

To be precise, let $f : G \to \mathbb{C}$. We describe the oracle as a function $O : G \to \mathbb{C}$ such that $O(x) = f(x)$ on the majority of $x \in G$. We assume that $\|O\|_\infty \leq \|f\|_\infty$. Define $R : G \to \mathbb{C}$ by $R(x) = O(x) - f(x)$ and let $I = \{x \in G : R(x) \neq 0\}$. We want to show that if $\widehat{f}(\alpha)$ is τ-heavy, then $\widehat{O}(\alpha)$ is τ'-heavy, for some τ' relatively large (its precise size depends on the success rate of the oracle).

Since $O = f + R$, then $\widehat{O}(\alpha) = \widehat{f}(\alpha) + \widehat{R}(\alpha)$. Note that $\|R\|_\infty \leq 2\|f\|_\infty$. Hence

$$\left|\widehat{O}(\alpha)\right| \geq \left|\widehat{f}(\alpha)\right| - \left|\frac{1}{|G|}\sum_{x \in I} R(x)\overline{\chi_\alpha}(x)\right| \geq \left|\widehat{f}(\alpha)\right| - \frac{2|I|}{|G|}\|f\|_\infty \ .$$

As I is small, if $\widehat{f}(\alpha)$ is significant then so is $\widehat{O}(\alpha)$. Note that as the reliability rate of the oracle decreases, so does the size of $\widehat{O}(\alpha)$, while the other coefficients increase in size. One can see that, similarly to majority rule, more samples are needed when the reliability rate of the oracle decreases.

It is well-known that the GL theorem finds the unique function in case of low noise rate, namely if the the noise rate is smaller than $\frac{1}{4} - \epsilon$. One immediately sees this from our analysis: the original function satisfies $|\widehat{f}(s)| = 1$, for the secret vector s, and so only one Fourier coefficient of O is larger than $\frac{1}{2}$.

3.4 Hardness of finding significant coefficients in the random access model

The SFT algorithm requires chosen queries. The aim of this section is to explain that one does not expect a general learning algorithm for problems where the function values cannot be chosen. Indeed, we will show that if such a learning algorithm existed then the learning parity with noise (LPN) and learning with errors (LWE) problems would be easy.

Recall the LPN problem: an instance is a list of samples $(a, b = \langle a, s\rangle + e(a)) \in \mathbb{Z}_2^n \times \mathbb{Z}_2$ for some secret value s and a function $e : \{0,1\}^n \to \{0,1\}$ which determines the noise. Define $\mathrm{LPN} : \{0,1\}^n \to \{0,1\}$ by $\mathrm{LPN}(a) := (-1)^b$. This is a 'noisy version' of the function $f(x) := (-1)^{\langle a,s\rangle}$ for which $\widehat{f}(s)$ is the only non-zero Fourier coefficient. For a small noise rate (as in LPN), as shown in Section 3.3, the coefficient $\widehat{\mathrm{LPN}}(s)$ is a significant coefficient for this function. Hence, if one could find significant coefficients in $\{0,1\}^n$ on random samples, then one could solve LPN given the samples (a, b). Since LPN is believed to be hard, one does not expect such a variant of the SFT algorithm to exist. Further evidence for the hardness of this problem in the random access model is that it is related to the problem of decoding a random binary linear code.

The same argument holds for LWE in $\mathbb{Z}_p^n$. In LWE one has samples $a \in \mathbb{Z}_p^n$ and $b = \langle a, s\rangle + e(a) \pmod{p}$ where $e(a)$ is "small" relative to p. Defining LWE$(a) := \omega_p^b$ one can show that the coefficient of the character $\chi_s(x) = \omega_p^{\langle x,s\rangle}$ is significant. Hence, if one could find the significant coefficients when given random samples, then one could solve LWE given the samples (a, b). Since we have good evidence that LWE is a hard problem, this shows that we do not expect to be able to learn significant Fourier coefficients in the random access model.

The modulus-dimension tradeoff for LWE [28] shows how to transform LWE in $\mathbb{Z}_p^n$ to LWE in $\mathbb{Z}_{p^d}^{n/d}$ (albeit with a different error distribution), and so one can conclude that finding significant coefficients in $\mathbb{Z}_{p^n}$ on random samples is at least as hard as solving LWE in $\mathbb{Z}_p^n$. This is an example of the connection between $\mathbb{Z}_2^n$ and $\mathbb{Z}_{2^n}$ as explained in Remark 8.

4. MODULUS SWITCHING

The SFT algorithm is considerably simpler to understand and implement for $\mathbb{Z}_2^n$ or $\mathbb{Z}_{2^n}$ than for $\mathbb{Z}_p$. Furthermore, for domains of size 2^n, considerable effort has been invested by researchers in the engineering community into making this algorithm more efficient with respect to various measures [17] (also see Mansour and Sahar [34]). Hence, it is natural to try to work with functions over $\mathbb{Z}_{2^n}$ instead of functions over $\mathbb{Z}_p$. We now sketch an approach that shows one can transform significant functions on $\mathbb{Z}_p$ into significant functions on $\mathbb{Z}_{2^n}$ where $2^n \approx p$. In analogy to similar ideas in lattice cryptography we call this "modulus switching".

These ideas are implicit in the work of Shor [41] on factoring with quantum computers. Shor extends a periodic function to a larger domain. The core idea is that if a function is periodic, then the period, which is a feature of the time domain, is preserved over any (large enough) domain. This fact is exploited by Shor, where his further ideas take place in the frequency domain. Shor's analysis provides a clear interaction between the representation of a (periodic) function in the time and frequency domains.

We extend these ideas to show that a much larger class of functions keeps the properties of their frequency domain representation, when extending their time domain. Specifically, significant coefficients are 'preserved' even when the time domain representation of the function is extended (here "preserved" means that there is a clear relation between the coefficient of the functions). We refer to Laity and Shani [27] for the technical details.

Let $N = 2^n > p$ be the smallest power of two greater than p. For a function $f : \mathbb{Z}_p \to \mathbb{C}$, we define

$$\widetilde{f}(x) := \begin{cases} f(x) & \text{when } 0 \le x < p, \\ 0 & \text{when } p \le x < N \end{cases} .$$

Note that the operation $f \mapsto \widetilde{f}$ is $\mathbb{C}$-linear. The basic observation (see Figure 1) is that for a character χ_α on $\mathbb{Z}_p$, $\widetilde{\chi}_\alpha(x)$ is a function on $\mathbb{Z}_N$ that is also concentrated.

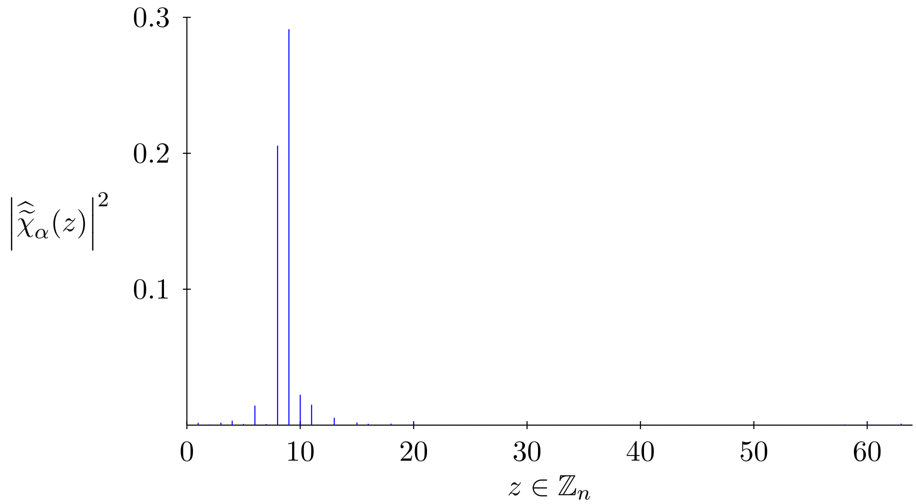

Figure 1: The magnitude of the Fourier coefficients $\widehat{\widetilde{\chi}}_\alpha(z)$. Here $p = 37$, $n = 64$ and $\alpha = 5$.

To explain this observation we state the following basic fact and sketch a proof of it. It is straightforward to turn this result into a rigorous upper bound.

Claim 10. *Let $N > 1$, $\omega_N = \mathrm{e}^{\frac{2\pi i}{N}}$ and let $\alpha \in \mathbb{R}$, $\alpha \neq 0$, $|\alpha| < N/2$ and $K \in \mathbb{N}$. Define*

$$S_{\alpha,K} = \sum_{x=0}^{K-1} \omega_N^{\alpha x}.$$

Then

$$|S_{\alpha,K}| \approx N \frac{|1 - \omega_N^{\alpha K}|}{2^{3/2}\pi|\alpha|}.$$

To see this note that the geometric series sums to $(1 - \omega_N^{\alpha K})/(1 - \omega_N^{\alpha})$ and the denominator is $(1 - \cos(2\pi\alpha/N)) - i\sin(2\pi\alpha/N)$ which has norm squared equal to $2(1 - \cos(2\pi\alpha/N))$. Finally, since $(1 - \cos(x)) \approx x^2/2$ (indeed $\frac{x^2}{2}(1 - \frac{x^2}{12}) \leq 1 - \cos(x) \leq \frac{x^2}{2}$) the result follows.

We now compute the Fourier transform of $\widetilde{\chi}_\alpha$ as a function on $\mathbb{Z}_N$ where $N = 2^n$. We have

$$\widehat{\widetilde{\chi}}_\alpha(\beta) = \langle \widetilde{\chi}_\alpha, \chi_\beta \rangle = \frac{1}{N} \sum_{x=0}^{p-1} \exp\left(2\pi i \left(\tfrac{\alpha}{p} - \tfrac{\beta}{N}\right) x\right).$$

If $\frac{\alpha}{p} - \frac{\beta}{N} \neq 0$, which will be satisfied in general since $\alpha, \beta \in \mathbb{Z}$ while $\gcd(p, N) = 1$, then applying Claim 10 gives the approximation

$$\widehat{\widetilde{\chi}}_\alpha(\beta) \approx \frac{1 - \exp(2\pi i(\alpha - \beta p/N))}{2^{3/2}\pi(\beta - \alpha N/p)}.$$

If $\beta \approx N\alpha/p$ then this coefficient is large and one sees that the function $\widetilde{\chi}_\alpha$ has a significant Fourier coefficient at $\lfloor N\alpha/p \rceil$. Moreover, the size of $\widehat{\widetilde{\chi}}_\alpha(\lfloor N\alpha/p + k \rceil)$ is proportional to $\frac{1}{|k|}$, and so $\widetilde{\chi}_\alpha$ is concentrated in a small set $\Gamma \subseteq \mathbb{Z}_N$ of characters near $N\alpha/p$.

Since the maps $f \mapsto \widetilde{f}$ and $g \mapsto \widehat{g}$ are $\mathbb{C}$-linear, for any $f(x) = \sum_{\alpha \in G} \widehat{f}(\alpha)\chi_\alpha(x)$ we have

$$\widehat{\widetilde{f}}(\beta) = \sum_{\alpha=0}^{p-1} \widehat{f}(\alpha)\widehat{\widetilde{\chi}}_\alpha(\beta).$$

Thus, if $\widehat{f}(\alpha)$ is a significant coefficient for f, then one expects that for $\beta = \lfloor N\alpha/p \rceil$, the coefficient $\widehat{\widetilde{f}}(\beta)$ is significant for $\widetilde{f}$. These arguments can be made more precise, and indeed can be extended to show that if $f(x)$ is a concentrated function on $\mathbb{Z}_p$ then $\widetilde{f}(x)$ is a concentrated function on $\mathbb{Z}_{2^n}$. We refer to [27] for the technical details.

As a consequence, one sees that it is not necessary to develop a variant of the SFT algorithm for the group $\mathbb{Z}_p$. Instead one can simply modulus-switch to a power of two and apply the SFT algorithm for the group $\mathbb{Z}_{2^n}$. Since the algorithms for $\mathbb{Z}_{2^n}$ have been optimised significantly (see [17, 34]) we believe that the resulting algorithms will be no less efficient than applying the AGS algorithm directly. Moreover, unlike the complexities working directly over $\mathbb{Z}_p$ as explained in Remark 9, this technique (although it might introduce new noise) overcomes the need to take overlapping intervals and is not subject to the choice of the interval.

The above discussion applies to extending a function from $\mathbb{Z}_p$ to $\mathbb{Z}_{2^n}$ where 2^n is slightly larger than p, and this was done by setting the new function to be zero on the new values. There are other ways to extend a function, for example using periodicity or some "intrinsic" description of the function (such as in the case of i-th bit functions). More generally one can consider modulus switching for domains of any size, including switching to a smaller domain. The results about concentration hold in this greater generality, and this provides a new technique to prove concentration of (some) families of functions, by showing that a sub-family, for domains of specific forms, is concentrated. We see an example of this in the next subsection.

4.1 The i-bit function is concentrated

We now explain that modulus switching provides an alternative proof of the Morillo-Ràfols result that every single-bit functions is concentrated [37]. We first explain that the i-th bit function on $\mathbb{Z}_{2^n}$ is concentrated. That is, $\{\mathrm{bit}_i : \mathbb{Z}_{2^n} \to \{-1, 1\}\}_{n \in \mathbb{N}}$ is concentrated.

Lemma 11. *Let $n \in \mathbb{N}$ and $0 \leq i < n$. Define $bit_i : \mathbb{Z}_{2^n} \to \{-1, 1\}$ by $bit_i(x) = (-1)^{x_i}$ where $x = \sum_{j=0}^{n-1} x_j 2^j$ and $x_j \in \{0, 1\}$. Let $\alpha \in \mathbb{Z}_{2^n}$. Then $\widehat{bit_i}(\alpha) = 0$ unless α is an odd multiple of 2^{n-i-1} in which case $|\widehat{bit_i}(\alpha)| = O(2^{n-i}/|\alpha|)$.*

Proof. Writing $N = 2^n$ we have $\widehat{\mathrm{bit}}_i(\alpha) = \frac{1}{N}\sum_{x=0}^{2^n-1} \mathrm{bit}_i(x)\omega_N^{-\alpha x}$. We write $x = y + 2^i b + 2^{i+1} z$ where $0 \le y < 2^i$, $0 \le z < 2^{n-(i+1)}$ and where b is the i-th bit. Then

$$\begin{aligned} \widehat{\mathrm{bit}}_i(\alpha) =& \frac{1}{N}\left(\sum_{y=0}^{2^i-1}\omega_N^{-\alpha y}\right)\left(\sum_{b=0}^{1}(-1)^b\omega_N^{-\alpha 2^i b}\right)\left(\sum_{z=0}^{2^{n-i-1}-1}\omega_N^{-2^{i+1}\alpha z}\right) \\ =& \frac{1}{N}\left(\sum_{y=0}^{2^i-1}\omega_N^{-\alpha y}\right)\left(1-\omega_N^{-2^i\alpha}\right)\left(\sum_{z=0}^{2^{n-i-1}-1}\omega_N^{-2^{i+1}\alpha z}\right) \quad . \end{aligned} \tag{9}$$

The third sum is just a sum over all 2^{n-i-1}-th roots of unity, so it is 2^{n-i-1} when α is a multiple of 2^{n-i-1} and otherwise is zero. The middle term $(1-\omega_N^{-2^i\alpha})$ is therefore 2 when α is an odd multiple of 2^{n-i-1} and is zero if it is an even multiple. Claim 10 can be applied to the first sum, showing it is bounded as $O(\frac{N}{\alpha})$.

The result then follows: the Fourier coefficient $\widehat{f}(\alpha)$ is zero when α is not an odd multiple of 2^{n-i-1} and when it is non-zero it has magnitude proportional to $2^{n-i}/|\alpha|$. ∎

The lemma shows that, when i is small there are a few non-zero coefficients (especially for $i = 0$, there is only one non-zero coefficient at $\alpha = 2^{n-i-1}$). When i is "medium" then there are non-zero coefficients at all multiples $\alpha = j2^{n-i-1}$, and they decrease in size with $1/j$. When i is large (e.g., $i = n-1$) then the significant coefficients are all close to 0 and are spaced at distance $2 \cdot 2^{n-1-i}$ (i.e., when $i = n-1$ they are 2 apart; for the second most significant bit they are spaced 4 apart, and so on). A corollary is that the i-th bit function on $\mathbb{Z}_{2^n}$ is concentrated.

Having established that the i-th bit function is concentrated on $\mathbb{Z}_{2^n}$, our modulus switching approach shows that the i-th bit function on $\mathbb{Z}_p$ is concentrated, since one can "switch down" from $\mathbb{Z}_{2^n}$ to $\mathbb{Z}_p$. This general approach gives a new and simpler proof of the result in [37] (the proof in [37] is very technical; they decompose $N = k2^i \pm m$ and consider different cases of m).

5. Applications in Cryptography

The SFT algorithm has been used to prove results about the hardness of recovering bits of Diffie-Hellman secrets keys in both finite fields and elliptic curves. It has also been used to reprove known results on the hardness of recovering bits of the secret values in the discrete logarithm problem (DLP) and RSA problem, and to prove some reductions for the learning with errors (LWE) [40] and learning with rounding (LWR) [6] problems. This section surveys how the SFT algorithm is used in these applications.

5.1 Background and motivation

A one-way function h, if it exists, assures that while given x it is easy to compute $h(x)$, retrieving x from $h(x)$ is hard. This hardness does not necessarily mean that given $h(x)$ one cannot find some partial information of x. Naturally, the main interest is in trying to learn some bits of x, but other sorts of partial information have also been considered. Bits of x that cannot be learnt from $h(x)$ are called *hardcore bits*. In other words, a hardcore bit is a bit which is as hard to compute as the entire secret value. For a historical overview see [19]. To show that a bit (or a set of bits) is hardcore, one usually tries to construct an algorithm that inverts h, given a target value $h(x)$ and an oracle that takes $h(t)$ and outputs a bit of t. In order to do so, one first needs to establish a way to query the oracle on values $h(t)$ such that there is some known relation between t and x, for example $t = \alpha x$ for known α's.

A useful language to describe these ideas is the *hidden number problem*, which was introduced by Boneh and Venkatesan [11] in order to study bit security of secrets keys arising from Diffie-Hellman key exchange. This problem turned out to be general enough to be applied to other cryptographic problems like DLP and RSA. In fact, the generality of the problem allows it to be used also outside of the scope of bit security (see [39, Section 4.4] and references within, also [13, 5]). Therefore, the hidden number problem is of theoretical interest and is studied today in its own right. It has many extensions and different variants; see [43] for a comprehensive survey.

Definition 12 (Hidden number problem)**.** Let $(G, \cdot)$ be a group, let $s \neq 0$ be a secret element of G and let f be a function defined over G. Find s using oracle access to the function $f_s(x) := f(s \cdot x)$.

We use the term *oracle access* as a general term for either of the following oracle models: in the *random access* model the solver receives polynomial many samples $(x, f_s(x))$ where the values x are drawn independently and uniformly at random from G; in the *query access* model the solver can query the oracle on any input $x \subseteq G$ and receive the answer $(x, f_s(x))$. To emphasize the difference between these models, we refer to the hidden number problem in the latter model as chosen multiplier hidden number problem (CM-HNP). This problem can also be divided into two models, namely *adaptive access* where the solver has a continuous access to the oracle and can query it at any time of the recovery process, and *non-adaptive access* where the solver is not allowed to query the oracle once the recovery process has started. Other types of access models could be also considered. For example, the original work on the hidden number problem [11] considers an oracle for which on the query $x \in \mathbb{Z}_p$ replies with $(x, f(sg^x))$.

An interesting case is when the oracle is unreliable. That is, the oracle does not give a correct answer all the time, but with some probability. It is common to call an oracle that always provides a correct answer a *perfect oracle*. An oracle that is correct only with some noticeable advantage is called an *unreliable* or *imperfect oracle*.

The following table summarizes some of the known results on the hidden number problem in different models. Here p is a prime number and 'imperfect' under the 'Oracle' column refers to an oracle with any non-negligible advantage over trivial guessing. The starting point of this work is the Boneh-Venkatesan result [11] which requires a perfect oracle and uses lattice methods rather than Fourier learning methods; this work was adapted to unreliable oracles by [20], but there is a complex tradeoff with the number of bits and so we do not include it in our table.

Problem	Access	Group	Bits	Oracle	Remarks
HNP	random	$\mathbb{Z}_p^*$	$\sqrt{\log p} + \log\log p$ MSB[6]	perfect	Given by [11]
CM-HNP	adaptive	$\mathbb{Z}_p^*$	LSB	imperfect	Given by [4]
CM-HNP	adaptive	$\mathbb{Z}_p^*$	any single bit	imperfect	Given by [23]
CM-HNP	non-adaptive	$\mathbb{Z}_N^*$	MSB & LSB	imperfect	Given by [8]
CM-HNP	non-adaptive	$\mathbb{Z}_N^*$	each single bit for the outer $\log\log p$ bits	imperfect	Given by [3]
CM-HNP	non-adaptive	$\mathbb{Z}_N^*$	any single bit	imperfect	Given by [37]

Most early works such as [4, 8, 23] require complicated algebraic manipulations such as tweaking and untweaking bits. Using the SFT algorithm [3] gives a uniform and clear approach. We present this solution to CM-HNP, using different terminology than the original one, for functions of norm 1, as the subsequent applications involve single bit functions (with the convention that $\text{bit}_i(x) = (-1)^{x_i}$ where x_i is the i-th bit of x).

Theorem 13 ([3]). *Let $f : \mathbb{Z}_N \to \{-1, 1\}$ be a function with a τ-heavy Fourier coefficient for $\tau^{-1} = poly(\log |G|)$. Then, the chosen multiplier hidden number problem in $\mathbb{Z}_N^*$ with the function f can be solved in polynomial time.*

In particular, the theorem holds for every concentrated function.

Remark 14 (Coding Theory terminology). *Theorem 13 rephrases Theorem 2 of [3]. The latter work gives a polynomial time list-decoding algorithm for* concentrated codes *with corrupted code words (Theorem 1) and subsequently a general list-decoding methodology for proving hardcore functions (Theorem 2). Most subsequent works on hardcore bits adopt this coding-theoretic language. Thus, in order to apply Theorem 2 of [3], these works use Theorem 1 of [3], which applies to concentrated codes. This caused these authors to put effort into proving that a particular code is concentrated. However, we emphasize that to apply the CM-HNP approach of [3] there is no need for the function to be concentrated. Instead it suffices that the function has a significant Fourier coefficient, and this is usually much easier to prove. We make this clear in our formulation of Theorem 13. In other words,*

[6]Since one can easily transform HNP with the LSB function to HNP with the MSB function, HNP can also be solved given $\sqrt{\log p} + \log\log p$ LSB. A generalization of this technique [38, Section 5.1] allows to transform HNP with $2d$ consecutive inner bits to HNP with d MSB, hence HNP can also be solved given $2(\sqrt{\log p} + \log\log p)$ consecutive inner bits.

while concentration is sufficient for a code to be recoverable *it is not a necessary condition. For these reasons (and others) we find the coding-theoretic language unhelpful and do not use it in this paper.*

We now sketch the proof of Theorem 13: run the SFT algorithm on f and f_s to get short lists L, L_s of τ-heavy coefficients for each function, respectively. By the scaling property $\widehat{f_s}(\alpha) = \widehat{f}(\alpha s^{-1})$ for every α. Therefore, for every $\beta \in L_s$ for which $\widehat{f_s}(\beta)$ is τ-heavy there exists $\alpha \in L$ such that $\beta = \alpha s^{-1}$. The secret s can be recovered efficiently. Notice that while the hidden number problem takes place in a multiplicative group, this solution involves Fourier analysis over an additive group.

A template for algorithms for CM-HNP is the following: show that (i) the "partial information" function f has a significant coefficient, (ii) the function f_s has a significant coefficient, and (iii) some (recoverable) relation between the coefficients of f and f_s exists. If one succeeds in showing these 3 conditions, then using the SFT algorithm one can solve this instance of CM-HNP. This template allows bit security researchers to look for settings where a solution to CM-HNP is already known (namely, cases where these 3 conditions are already known to hold, like single-bit functions over $\mathbb{Z}_N$) and try to convert their problem of interest to this setting.

5.1.1 The multivariate hidden number problem

Another case of interest is the *multivariate hidden number problem* (MVHNP), which we define as follows.

Definition 15 (Multivariate hidden number problem)**.** Let R be a ring, let $\mathbf{s} = (s_1, \ldots, s_m) \neq (0, \ldots, 0)$ be a secret element in R^m and let f be a function defined over R. Find $\mathbf{s}$ using oracle access to the function $f_{\mathbf{s}}(\mathbf{x}) := f(\mathbf{s} \cdot \mathbf{x}) = f(s_1 x_1 + \cdots + s_m x_m)$.

Specific instances of this problem are LWE and LWR.[7] This problem is also related to bit security of Diffie-Hellman secrets in (finite) extension fields [16] (and similarly, to the polynomial version of LWE).

Similar to the solution to HNP in $\mathbb{Z}_p$, one expects to have a solution in $\mathbb{Z}_{p^m}$ for the $\sqrt{\log(p^m)} = \sqrt{m \log(p)}$ MSB's. A result in this fashion is given by Shparlinski in [42] to the *polynomial HNP*. As mentioned in the final two sentences of [42], this result also holds for general multivariate polynomials. However since MVHNP consists of a multivariate polynomials of degree one, it is possible to adjust this solution even when only $\sqrt{\log(p)}$ MSB's are given. This however, requires p not polynomial in m.

The resulting algorithm is a rather straightforward lattice attack, very similar to known algorithms for LWE, that reduces the problem to the closest vector problem in a certain lattice. A related problem is the *trace HNP* [30], which can be interpreted as MVHNP under some basis, due the linearity of trace.

[7] LWR can be interpreted as giving MSB's of the inner product.

One can also define a chosen-multiplier version of the multivariate hidden number problem (CM-MVHNP), as done in [16]. By proving an analogue of the Fourier scaling property in higher dimensions one can generalize Theorem 13 to the case of CM-MVHNP.

5.2 Applications

We present some of the applications in cryptography of the SFT algorithm. They are all based on reducing some problems to the CM-HNP or CM-MVHNP. In the following we assume to have an oracle O that solves some problem, and show how to use this oracle to solve a harder problem, thus establishing the hardness equivalence between the two problems.

5.2.1 Proving known results: bit security of RSA and DLP

The first application of the algorithm was given in [3], where it is shown that the most-significant bit and least-significant bit are hardcore for the RSA function $RSA_{N,e}(x) := x^e \pmod N$ and for exponentiation $EXP_g(x) := g^x$, where g is an element of prime order ℓ in some group. The results hold for imperfect oracles that have noticeable advantage over guessing. These results were already known, as [4] first shows that the LSB is hardcore for the RSA function and [23] shows that every bit is hardcore for both functions. Nevertheless, the approach based on SFT is more general (holds for every function with significant coefficients) and simpler. We explain how to derive these results.

Given an instance $RSA_{N,e}(x) = x^e \pmod N$, we want to recover x. Since the values e, N are public in the RSA setting, for every number r one can calculate $RSA_{N,e}(rx \pmod N)$ by $(r^e \pmod N)(x^e \pmod N) = (rx)^e \pmod N$. Hence, given $RSA_{N,e}(x)$ one can query the oracle on $RSA_{N,e}(rx)$ to get a bit of rx for every chosen r. The problem therefore becomes the CM-HNP in $\mathbb{Z}_N^*$, and this can be solved using the SFT algorithm over the additive group $(\mathbb{Z}_N, +)$, which has known order.

Similarly, given an instance $EXP_g(x) = g^x$, since g and ℓ are public, one can calculate $EXP_g(rx \pmod \ell)$ for every number r by $(g^x)^r = g^{rx}$. Thus, given $EXP_g(x)$ one can query the oracle on $EXP_g(rx)$ to get a bit of rx for every chosen r. The problem therefore becomes the CM-HNP in $\mathbb{Z}_\ell^*$, and this can be solved using the SFT algorithm over the additive group $(\mathbb{Z}_\ell, +)$. This proves bit security results for the DLP in finite fields and elliptic curves.

Applying Theorem 13 we find that all bits for those functions are hardcore. This result also holds for other functions, as *Rabin* (see [1, Chapter 7]) and the Paillier trapdoor permutation (see [37, Section 7]).

5.2.2 Bit security of the Diffie-Hellman protocol and related schemes

An open question is to prove that single bits of Diffie-Hellman keys are hardcore. Here we assume that we have an oracle O that on g, g^a, g^b returns a single bit of the Diffie-Hellman key $s = g^{ab}$. To interact with the oracle, notice that given g^b one can calculate $g^{b+r} = g^b g^r$ for any number r. One can then query the oracle O with g, g^a, g^{b+r} and receive a bit of $g^{a(b+r)} = g^{ab}g^{ar} = st$. This is how the hidden number problem was originally identified. This application does not correspond to the CM-HNP, since choosing the multiplier t is equivalent to finding discrete logarithms for the base g^a in $\mathbb{Z}_p^*$.

For related schemes where the exponent is fixed, Akavia [2] followed Boneh-Venkatesen [12] to get around this problem by assuming an "advice" that provides the discrete logarithms of the chosen multipliers t to the base g^a, but this is not realistic in actual applications (see also our remark in Section 6.2). There is currently no method known to prove the hardness of single bits of Diffie-Hellman keys in the usual model.

To overcome this problem, Boneh and Shparlinski [10] suggested (in the context of elliptic curves) a different model where the oracle O takes as input a group homomorphism $\phi : G \to G'$, the values g, g^a, g^b, and then outputs $O(\phi(g^{ab}))$. The approach is then to keep the inputs g, g^a, g^b fixed and to use ϕ as the way to choose multipliers in the hidden number problem. We call this the *representation changing model*. This model allows to convert the nonlinear Diffie-Hellman problem to an easier linear problem.

In this case one can think of s as some secret element (not necessarily a Diffie-Hellman key as the interaction with the oracle does not come from the key exchange setting), for which one receives $O(\phi(s))$. Now assume s has several components, $\mathbf{s} = (s_1, \ldots, s_n)$, such that the oracle O returns a bit of some component $\phi(s)_i$. In this model, if for some $1 \le i \le n$ the family of functions $\{\phi_i^r(\mathbf{x}) := rx_i\}$ is a family of isomorphisms,[8] then getting a single bit of $\phi_i^r(\mathbf{g^{ab}}) = r(g^{ab})_i = rs_i$ for chosen r, gives rise to CM-HNP with single-bit functions. Therefore, one can show hardness of single bits in this model, if one can find a group for which the condition on the functions holds (note that one only recovers a component of s, and therefore needs other methods for recovering the entire value s; for the case in which $\mathbf{s} = \mathbf{g^{ab}}$ is a Diffie-Hellman key in $\mathbb{F}_{p^m}$ that we describe below, one can use the results involving "summing functions" from [47] and recover the entire secret $\mathbf{s}$ from the algorithm that recovers a single (fixed) component s_i).

As mentioned above, this idea was introduced by Boneh and Shparlinski [10] for the LSB of (each coordinate of) Diffie-Hellman secrets in elliptic curve groups over prime fields. It is shown that changing the Weierstrass equation is an isomorphism that gives rise to the desired functions. They then use the same technique as in [4] to prove hardness of LSB. This approach was then applied by [14] to every

[8] It is sufficient that there is a 'large enough' subfamily of isomorphisms.

single bit of Diffie-Hellman secret keys in elliptic curves, using the SFT algorithm (that is, using the solution to CM-HNP for single-bit functions).

The idea of changing group representations can also be used for finite fields. The works [15, 48] consider the computational Diffie-Hellman problem in groups $\mathbb{F}_{p^m}^*$ for $m > 1$. They show that some polynomial representations give rise to the desired functions, and therefore reduce to CM-HNP.

For a detailed overview of these techniques we refer the reader to the exposition of Sections $5, 5.1, 5.2$ and subsections within of [16]. This latter work gives applications of the solution for CM-MVHNP to show bit security of the computational Diffie-Hellman problem in groups of higher dimension in models similar to those mentioned above; specifically, for elliptic curves over extension fields, and for $\mathbb{F}_{p^m}^*$ with different representations of the field $\mathbb{F}_{p^m}$.

We stress that these models do not tell a lot about the hardness of specific bits in real life implementations of Diffie-Hellman key exchange, where the representation of the group is fixed. One should interpret results in the representation changing model as follows: assuming hardness of CDH in a group G (where G can be the multiplicative group of a finite extension field or an elliptic curve over a finite field), there is no algorithm that takes $g, g^a, g^b \in G$ and outputs the i-th bit of g^{ab} for many representations of G (more precisely, for representations corresponding to the specific isomorphisms used in the reduction). Nevertheless, given an instance g^a, g^b in a specific representation of G, this result does not tell us whether it is hard to compute a specific bit of the secret g^{ab}. Indeed, this problem is still open.

5.2.3 Sample-preserving search-to-decision reductions for LWE and LWR

We assume the reader is familiar with the *search* and *decision* variants of the LWE and LWR problems. We only focus on the part of the reduction which involves the SFT algorithm; the entire reduction is more involved. By a "hybrid" argument (see [19, Theorem 1] or [9, Lemma 3]), one can reduce the decision problem to distinguishing a single LWE sample.[9] We therefore consider a single LWE sample. The standard method to show that the decision problem is as hard as the search problem is as follows. Suppose one has a perfect decision oracle. Given an LWE sample $b = \langle \mathbf{a}, \mathbf{s} \rangle + e = a_1 s_1 + \ldots + a_n s_n + e \pmod{p}$ one makes a guess s' for s_1 and re-randomises the sample as $\mathbf{a}' = (a_1 + r, a_2, \ldots, a_n), b' = b + rs' \pmod{p}$. If the guess is correct (i.e., if $s' = s_1$) then $(\mathbf{a}', b')$ is a valid LWE sample whereas if the guess is incorrect then b is uniform. Hence the decision oracle determines whether the guess of the secret is correct. After at most pn queries to the decision oracle one can compute the secret.

When the oracle is not perfect one will have to repeat this procedure with different inputs $\mathbf{a}$ and follow majority rule. When the success rate of the oracle is low, one may not have enough initial inputs $\mathbf{a}$ to satisfactorily apply the majority rule, and therefore would need to draw more samples. A *sample-preserving* reduction is a reduction that uses only the initial given samples, and does not ask

[9]The reduction given in [35] uses the duality of the LWE and knapsack functions is used for this result.

for more samples during the procedure. Micciancio and Mol [35] used the SFT algorithm to show a sample-preserving search-to-decision reduction for the learning with errors problem. We explain this reduction.

The standard method involves choosing a unit vector $\mathbf{u}_1 = (1, 0, \ldots, 0)$ and guessing $\langle \mathbf{u}_1, \mathbf{s}\rangle$. Micciancio and Mol observe that one can choose any random vector $\mathbf{v}$ and guess $\langle \mathbf{v}, \mathbf{s}\rangle$, then let the decision oracle to advise whether this guess is correct or incorrect. Notice that one can try all possible p guesses for the same value $\langle \mathbf{v}, \mathbf{s}\rangle$, and store the one on which the oracle replied that the guess is correct, or keep drawing new vectors $\mathbf{v}$ and make only one guess for $\langle \mathbf{v}, \mathbf{s}\rangle$, denoted by $b_{\mathbf{v}}$. The latter approach is taken in [35], where if the oracle says that the guess for $\langle \mathbf{v}, \mathbf{s}\rangle$ is incorrect (more precisely, that the distribution is uniform), then one takes $b_{\mathbf{v}}$ to be any value from the remaining $p - 1$ possibilities.

Again, if the oracle is perfect then one determines the correct guesses and eventually obtains n linear equations in $\mathbf{s}$ and hence can solve the problem. However if the oracle is not perfect (but has a noticeable advantage over a random guess), then for a selection of chosen vectors $\mathbf{v}$ we have the values $b_{\mathbf{v}}$, for which $b_{\mathbf{v}} = \langle \mathbf{v}, \mathbf{s}\rangle$ with some noticeable bias from $\frac{1}{p}$. In other words, we have query access to a noisy version of the function $f(\mathbf{v}) = \langle \mathbf{v}, \mathbf{s}\rangle \pmod p$. This is an instance of CM-MVHNP with an unreliable oracle. One runs the SFT algorithm on the function $\omega_p^{b_{\mathbf{v}}}$, which is a noisy version of $\omega_p^{\langle \mathbf{v}, \mathbf{s}\rangle}$, to find the significant coefficient, hence the character, and thus solve this problem.

A very similar approach is taken in [9] for the learning with rounding problem. We remark that the reduction is an average-case reduction, and does not hold for worst case. A sample-preserving reduction for the latter is still an open problem.

6. Limitations

The solution to the CM-HNP in $\mathbb{Z}_N$ (Theorem 13) is based on Fourier analysis in the additive group $(\mathbb{Z}_N, +)$ and it exploits the scaling property of the Fourier transform for the function $f_s(x) := f(sx)$. In other words, the function f_s is the composition of f with a linear map on $\mathbb{Z}_N$. It is natural to consider whether this approach can be used for other algebraic groups (such as elliptic curves and algebraic tori). The hidden number problem in the case of elliptic curves is to determine a secret point $S \in E(\mathbb{F}_p)$ given samples $(P, f(S+P))$ where a typical choice for the function would be $f(Q) = \mathrm{bit}_i(x(Q))$. The natural approach is to still use Fourier analysis in the additive group $(\mathbb{Z}_p, +)$ but instead of composing with a linear map, to compose with a rational function (e.g., coming from the translation map $t_S(P) = P+S$). Another generalisation would be Fourier analysis in other groups $(G, \cdot)$.

If such tools could be developed we might have an approach to the bit security of Diffie-Hellman key exchange in the group of elliptic curve points in certain models. There are also other interesting problems that could be approached with Fourier analysis on general groups. For example, the authors of [31] raise the question whether it is possible to apply these results to the *modular inversion hidden number problem*.

Unfortunately, there is a major obstacle to applying the SFT algorithm to these sorts of problems. Namely, if f is a concentrated function then the composition $f \circ \varphi$ is concentrated only when φ is affine. The aim of this section is to explain this obstacle. Since the translation map for the elliptic curve group law is a non-linear rational function, this explains why the method cannot be directly applied to the elliptic curve hidden number problem. Our argument also answers the question of [31] in the negative.

Let $f : G \to \mathbb{C}$ be a function and let $f_s(x) = f \circ \varphi_s(x)$, where $\varphi_s : G \to G$ is an efficiently computable function (that depends on some unknown value s). To generalise the proof of Theorem 13 one needs the following three conditions:

1. the function f has significant coefficients;
2. the function f_s has significant coefficients;
3. there exists a relation between the significant coefficients of f and f_s that allows to determine s (or at least a small set of candidates for s).

One special case is when f is a constant function. Then f_s is also a constant function and both conditions 1 and 2 are satisfied. The problem is that a constant function cannot tell us anything about the secret s, and so condition 3 does not hold. Hence, we need to focus on functions that are far from constant, which we formalise in our proof by requiring that $\widehat{f}(0) = 0$ (in other words, f is "balanced").

Having dispensed with this special case we focus on the first two conditions. We first consider the case when f is concentrated. If $\varphi_s(x) = ax + b$ is affine then we already know from the scaling and time-shifting properties that all Fourier coefficients of f are preserved, and so if f is concentrated then f_s is also concentrated. Our aim is to show a converse to this fact: if φ_s is a rational function and if conditions 1 and 2 both hold then φ_s must be affine. This result is closely related to the Beurling-Helson Theorem [7] (see [26, 29] for related results in $\mathbb{Z}_p$) and the work of Green and Konygin [22] on the Fourier transform of balanced functions.

For our result we need the following lemma [38, Lemma 7] (a proof, for general fields $\mathbb{F}_{p^m}$, can be found in [36, Theorem 2]).

Lemma 16. *Let q be prime. For any polynomials $f, g \in \mathbb{F}_q[x]$ such that the rational function $h = \frac{f}{g}$ is not constant in $\mathbb{F}_q$, the following bound holds*

$$\left| \sum_{\lambda \in \mathbb{F}_p} {}^{*} \omega_q^{h(\lambda)} \right| \leq (\max\{\deg(f), \deg(g)\} + u - 2)\sqrt{q} + \delta \ ,$$

where $\sum^*$ *means that the summation is taken over all* $\lambda \in \mathbb{F}_q$ *which are not poles of* h *and*

$$(u, \delta) = \begin{cases} (v, 1) & if\ \deg(f) \leq \deg(g), \\ (v+1, 0) & if\ \deg(f) > \deg(g), \end{cases}$$

and v *is the number of distinct zeros of* g *in the algebraic closure of* $\mathbb{F}_q$.

We formulate the following result for functions on $\mathbb{Z}_q$ for a prime q, but it can be generalised to finite fields $\mathbb{F}_{p^m}$ with $m > 1$. Let $g, h \in \mathbb{Z}_q[x]$ be polynomials where h is not the constant zero. Let Z_h be the set of zeroes in $\mathbb{Z}_q$ of h. We define $\varphi(x) = g(x)/h(x)$ for all $x \in \mathbb{Z}_q \setminus Z_h$ and $\varphi(x) = 0$ otherwise (since we will assume Z_h is small compared with q it does not matter how we define φ on Z_h).

Recall that the definition of concentration applies to families of functions. To keep the formulation of the following proposition clean, we call a single function concentrated as explained after the definition above.

Proposition 17. *Let* q *be a sufficiently large prime. Let* f *be a concentrated function on* $\mathbb{Z}_q$ *such that* $\|f\|_2 = 1$ *and* $\widehat{f}(0) = 0$. *Let* $g, h \in \mathbb{Z}_q[x]$ *be polynomials of degree bounded by* $poly(\log(q))$ *and let* Z_h *be the set of zeroes of* h. *Define* $\varphi(x)$ *as above and suppose this function is non-constant. Let* $\tau = 1/poly(\log(q))$. *If* $f \circ \varphi$ *has any* τ*-heavy Fourier coefficients then* $\varphi(x) = ax + b$ *for some* $a, b \in \mathbb{Z}_q$.

Proof. Let $G = \mathbb{Z}_q$ and write $f = \sum_{\alpha \in G} \widehat{f}(\alpha)\chi_\alpha$. Let $d = \max\{\deg(g(x)), \deg(h(x))\}$. For contradiction we suppose $\varphi(x) \neq ax + b$ for any a, b. Let $\epsilon = \frac{\tau}{32d^2}$. Since f is concentrated there is a set Γ of size $poly(\log(|G|))$ such that

$$\|f - f|_\Gamma\|_2^2 \leq \epsilon = \frac{\tau}{32d^2}.$$

Since $\widehat{f}(0) = 0$ it follows that Γ does not contain zero.

Now consider $f_\varphi(x) = f(\varphi(x)) = \sum_{\alpha \in G} \widehat{f}(\alpha)\chi_\alpha(\varphi(x))$. For every $\beta \in G$ we have

$$\begin{aligned} \widehat{f_\varphi}(\beta) = & \frac{1}{|G|}\sum_{x \in G} f_\varphi(x)\overline{\chi_\beta(x)} = \frac{1}{|G|}\sum_{x \in G} f(\varphi(x))\overline{\chi_\beta(x)} = \\ & \frac{1}{|G|}\sum_{x \in G}\sum_{\alpha \in G} \widehat{f}(\alpha)\chi_\alpha(\varphi(x))\overline{\chi_\beta(x)} = \frac{1}{|G|}\sum_{\alpha \in G} \widehat{f}(\alpha)\sum_{x \in G}\chi_\alpha(\varphi(x))\overline{\chi_\beta(x)} = \\ & \frac{1}{|G|}\sum_{\alpha \in G} \widehat{f}(\alpha)\sum_{x \in G}\chi_1(\alpha\varphi(x) - \beta x) = \frac{1}{|G|}\sum_{\alpha \in G} \widehat{f}(\alpha)\sum_{x \in G}\chi_1(\psi_\alpha^\beta(x)) \ , \end{aligned}$$

where we denote $\psi_\alpha^\beta(x) = \alpha\varphi(x) - \beta x$. Since $\widehat{f}(0) = 0$ we can ignore the case $\alpha = 0$ and by our supposition that $\varphi \neq ax + b$ we know that there are no α, β such that ψ_α^β is constant. Hence, the last sum is a character sum satisfying the conditions of Lemma 16. Furthermore, $\psi_\alpha^\beta = (\alpha g(x) - \beta x h(x))/h(x)$

and so the value u in Lemma 16 is bounded by $\max\{\deg(g), \deg(h)\} \le d$. Applying Lemma 16, we get that for every $\alpha \neq 0$ and every β it holds that $|\sum_{x\in G\setminus Z_h} \chi(\psi_\alpha^\beta(x))| \le C$ where $C = 2d\sqrt{q}$.

Now note that

$$\widehat{f_\varphi}(\beta) = \frac{1}{|G|}\sum_{\alpha\in G}\widehat{f}(\alpha)\sum_{x\in Z_h}\chi_1(\psi_\alpha^\beta(x)) + \frac{1}{|G|}\sum_{\alpha\in\Gamma}\widehat{f}(\alpha)\sum_{x\in G\setminus Z_h}\chi_1(\psi_\alpha^\beta(x)) + \frac{1}{|G|}\sum_{\alpha\notin\Gamma}\widehat{f}(\alpha)\sum_{x\in G\setminus Z_h}\chi_1(\psi_\alpha^\beta(x)) \ .$$

For the first term we note that $|\sum_{x\in Z_h}\chi_1(\psi_\alpha^\beta(x))| \le |Z_h| \le d$ and that $\|f\|_2 = 1$ implies $\sum_{\alpha\in G}|\widehat{f}(\alpha)| \le \sqrt{|G|} = \sqrt{q}$ and $|\widehat{f}(\alpha)| \le 1$ for all α. Therefore

$$\left|\widehat{f_\varphi}(\beta)\right| \le \frac{d}{\sqrt{q}} + \left|\frac{1}{|G|}\sum_{\alpha\in\Gamma}\widehat{f}(\alpha)\sum_{x\in G\setminus Z_h}\chi(\psi_\alpha^\beta(x))\right| + \left|\frac{1}{|G|}\sum_{\alpha\notin\Gamma}\widehat{f}(\alpha)\sum_{x\in G\setminus Z_h}\chi(\psi_\alpha^\beta(x))\right| \ .$$

We apply the triangle inequality on the first sum and the Cauchy-Schwarz inequality on the second. Let $k = |\Gamma|$ and write $\Gamma = \{\alpha_1, \ldots, \alpha_k\}$ and $q = |G|$. Then using Lemma 16 we get

$$\begin{aligned}\left|\frac{1}{|G|}\sum_{\alpha\in\Gamma}\widehat{f}(\alpha)\sum_{x\in G\setminus Z_h}\chi(\psi_\alpha^\beta(x))\right| &= \left|\frac{1}{|G|}\sum_{j=1}^{k}\widehat{f}(\alpha_j)\sum_{x\in G\setminus Z_h}\chi(\psi_{\alpha_j}^\beta(x))\right| \le \left|\frac{1}{q}\sum_{j=1}^{k}\widehat{f}(\alpha_j)\cdot C\right| \\ &\le \frac{1}{q}\sum_{j=1}^{k}\left|\widehat{f}(\alpha_j)\right| C = \frac{2kd}{\sqrt{q}} \ .\end{aligned}$$

Since $k = |\Gamma| = poly(\log(q))$ we have that this bound (similarly for the earlier bound $d/\sqrt{q}$) is negligible, so we have for example

$$\frac{d}{\sqrt{q}} + \frac{2kd}{\sqrt{q}} < 2d\sqrt{\epsilon}.$$

From Parseval's identity $\sum_{\alpha\notin\Gamma}\left|\widehat{f}(\alpha)\right|^2 = \|f - f|_\Gamma\|_2^2 \le \epsilon$. Therefore, by the Cauchy-Schwarz inequality we have

$$\left|\frac{1}{|G|}\sum_{\alpha\notin\Gamma}\widehat{f}(\alpha)\sum_{x\in G\setminus Z_h}\chi(\psi_\alpha^\beta(x))\right| \le \frac{1}{|G|}\left(\sum_{\alpha\notin\Gamma}\left|\widehat{f}(\alpha)\right|^2\right)^{\frac{1}{2}}\left(\sum_{\alpha\notin\Gamma}\left|\sum_{x\in G\setminus Z_h}\chi(\psi_\alpha^\beta(x))\right|^2\right)^{\frac{1}{2}} \le \frac{1}{|G|}\sqrt{\epsilon}\left(\sum_{\alpha\notin\Gamma}C^2\right)^{\frac{1}{2}} \ .$$

Then

$$\left|\frac{1}{|G|}\sum_{\alpha\notin\Gamma}\widehat{f}(\alpha)\sum_{x\in G\setminus Z_h}\chi(\psi_\alpha^\beta(x))\right| \le \frac{\sqrt{\epsilon}\sqrt{q-k}2d\sqrt{q}}{q} \le 2d\sqrt{\epsilon} \ .$$

Finally, combining the two bounds we get

$$\left|\widehat{f_\varphi}(\beta)\right|^2 \le \left(\frac{d}{\sqrt{q}} + \frac{2kd}{\sqrt{q}} + 2d\sqrt{\epsilon}\right)^2 < \left(4d\sqrt{\epsilon}\right)^2 = \left(4d\frac{\sqrt{\tau}}{4d\sqrt{2}}\right)^2 = \frac{\tau}{2} \ .$$

Therefore, for every β the coefficient $\widehat{f_\varphi}(\beta)$ is not τ-heavy for any noticeable τ.This gives the required contradiction and so we conclude that φ is affine. ■

6.1 ϵ-concentrated functions

Proposition 17 shows that if f is concentrated (and far from constant) and $f \circ \varphi$ has significant coefficients, then φ is affine. It is natural to wonder whether the condition that f is concentrated is necessary. In fact, the result cannot be weakened in general: if $\varphi(x) = g(x)/h(x)$ is non-affine and invertible almost everywhere (such as a Möbius function $\varphi(x) = (ax+b)/(cx+d)$ where $ad - bc = 1$) then $f(x) = \chi_\alpha(x) + \chi_\beta(\varphi^{-1}(x))$ is such that $f(x)$ has a significant coefficient at α and $f \circ \varphi$ has a significant coefficient at β.

However, a version of Proposition 17 is true for some non-concentrated functions of interest. Since Theorem 13 does not require the function to be concentrated, it is of interest to also show that composing with non-affine $\varphi(x)$ is an obstruction to the solution to CM-HNP for these functions as well. Hence, for the rest of this section we consider a 'noisy character', $f(x) := \omega_N^{\alpha x + e(x)}$. We first show that these functions have a significant coefficient, then we show that $f \circ \varphi$ does not have a significant coefficient when φ is not affine.

To formalise the problem we think of $e(x)$ as a random variable from some distribution (e.g., a discrete Gaussian or a uniform distribution on some small interval compared with N). We treat $e(x)$ as being independent of x, in which case we can write

$$\widehat{f}(\beta) = \mathbb{E}\left(\omega_N^{\alpha x - \beta x + e(x)}\right) = \mathbb{E}\left(\omega_N^{(\alpha-\beta)x}\omega_N^{e(x)}\right) = \mathbb{E}\left(\omega_N^{(\alpha-\beta)x}\right)\mathbb{E}\left(\omega_N^{e(x)}\right).$$

To show that $|\widehat{f}(\alpha)|$ is large it suffices to give a lower bound for $\left|\mathbb{E}\left(\omega_N^{e(x)}\right)\right|$. We do this by following an argument due to Bleichenbacher [8].

Bleichenbacher defines the *bias* of a random variable X on $\mathbb{Z}$ as

$$B_N(X) = \mathbb{E}\left(\exp(2\pi i X/N)\right) \quad .$$

Assume X is the uniform distribution in some interval $[0, T-1]$ for some $0 < T \leq N$. Then

$$B_N^U(X) := B_N(X) = \frac{1}{T}\sum_{0 \leq x < T} \exp(2\pi i x/N) \quad .$$

Some properties of $B_N^U(X)$ appear in Lemma 1 of [13]. Since the latter is a geometrical progression,

$$B_N^U(X) = \frac{1}{T}\frac{\sin(\pi T/N)}{\sin(\pi N)} \quad .$$

Suppose $e(x)$ follows the uniform distribution X. That is, for each $x \in \mathbb{Z}_N$ the value $e(x)$ is chosen uniformly and independently at random in $[0, T-1]$. From linearity it is easy to see that

$$\mathbb{E}\left(\omega_N^{e(x)}\right) = \frac{1}{N}\sum_{x \in \mathbb{Z}_N}\left(\exp(2\pi i e(x)/N)\right) = \frac{N/T}{N}\sum_{0 \leq t < T}\exp(2\pi i t/N) = \frac{1}{T}\sum_{0 \leq t < T}\exp(2\pi i t/N) = B_N^U(X) \quad .$$

It is obvious that if $T = N$ then $B_N^U(X) = 0$. In applications $e(x)$ usually represents some given bits, and so it is natural to restrict $T \leq N/2$ as we do, though the following argument also holds given a fraction of a bit. For $T \leq N/2$ one has[10] $|B_N^U(X)| > 0.5$, and so $\left|\mathbb{E}\left(\omega_N^{e(x)}\right)\right|^2 = |B_N^U(X)|^2 > 0.25$. The desired lower bound is provided.

A similar approach holds when e follows a Gaussian distribution. In this case the size of the bias is even larger, as $e(x) = 0$ on a large set (and $e(x)$ is small on an even larger set) and so most of the energy is distributed around zero.

Hence, we have established that a noisy character has a significant coefficient. Finally, we address the result of Proposition 17 for such a function.

Claim 18. *Let φ be as in Proposition 17, and let $e(x)$ given by the uniform distribution (over some interval in $\mathbb{Z}_N$) or by a Gaussian distribution. If $f_\varphi(x) := \omega_N^{\varphi(x)+e(x)}$ has a significant coefficient then $\varphi(x) = ax + b$ for some $a, b \in \mathbb{Z}_N$.*

Proof. We observe that for every β

$$\widehat{f_\varphi}(\beta) = \mathbb{E}\left(\omega_N^{\varphi(x)-\beta x+e(x)}\right) = \mathbb{E}\left(\omega_N^{\psi_1^\beta(x)}\omega_N^{e(x)}\right) = \mathbb{E}\left(\omega_N^{\psi_1^\beta(x)}\right)\mathbb{E}\left(\omega_N^{e(x)}\right) \quad ,$$

where $\psi_1^\beta(x) = \varphi(x) - \beta x$. Since $\left|\mathbb{E}\left(\omega_N^{e(x)}\right)\right| \leq 1$, it suffices to upper-bound $\left|\mathbb{E}\left(\omega_N^{\psi_1^\beta(x)}\right)\right|$. Such a bound follows from Lemma 16 in the same way as in the proof of Proposition 17. ∎

6.2 Hidden number problem in subgroups

Another limitation on the applications of the SFT algorithm is the following. Suppose that the multipliers in the hidden number problem are drawn from some set $H \subseteq G$. One can consider the multipliers to be in a proper subgroup $H < G$, as done in [21, 44]. It is not clear how to apply the SFT algorithm to solve this variant of the (chosen multiplier) hidden number problem. Specifically, the chosen queries in the algorithm have to be correlated, but it is not guaranteed that these correlated queries will all lie in the same subgroup. If the index $[G : H]$ is small (e.g., $[G : H] = 2$, as in the case of the set of squares in $\mathbb{F}_p^*$) then the issue can be managed, but if $[G : H]$ is large then no results are known. Therefore, for results (on Diffie-Hellman related schemes) that rely on advice of the form of discrete logarithms to some base g (as in [2, 12, 45]), if g generates a relatively small subgroup, it is not guaranteed that the desired correlated multipliers are indeed in the group generated by g. This restricts, for example, the result given in [2, Section 5]. This observation is similar to the one in [43, Section 2.5], and was handled in [10, Section 5] and [14, 4.1] since the set of squares in $\mathbb{F}_p^*$ has index 2 in $\mathbb{F}_p^*$.

[10] See [13, Table 1] for some values $|B_N^U(X)|$ for different $T \leq N/2$.

Acknowledgements

We thank Ben Green for providing some insights and references.

References

[1] Akavia, A. (2008) "Learning Noisy Characters, Multiplication Codes and Hardcore Predicates." Ph.D. Thesis. Massachusetts Institute of Technology.

[2] Akavia, A. (2009) "Solving Hidden Number Problem with One Bit Oracle and Advice," in S. Halevi (ed.), CRYPTO 2009, Springer LNCS 5677, 337–354.

[3] Akavia, A., Goldwasser, S., and Safra, S. (2003) "Proving Hard-Core Predicates Using List Decoding," in FOCS 2003, 146–157.

[4] Alexi, W., Chor, B., Goldreich, O., and Schnorr, C.P. (1988) "RSA and Rabin Functions: Certain Parts are as Hard as the Whole," in SIAM Journal on Computing, **17(2)**, 194–209.

[5] Aranha, D.F., Fouque, P.-A., Gérard B., Kammerer, J.-G., Tibouchi., M, and Zapalowicz, J.-C. (2014) "GLV/GLS Decomposition, Power Analysis, and Attacks on ECDSA Signatures with Single-Bit Nonce Bias," in P. Sarkar and T. Iwata (eds.) ASIACRYPT 2014. Springer LNCS 8873, 262–281.

[6] Banerjee, A., Peikert, C., and Rosen, A. (2012) "Pseudorandom Functions and Lattices," in D. Pointcheval and T. Johansson (eds.), EUROCRYPT 2012, Springer LNCS 7237, 719–737.

[7] Beurling, A., and Helson, H. (1953) "Fourier-Stieltjes Transforms with Bounded Powers," in Math. Scand., 1, 120–126.

[8] Bleichenbacher, D. (2000) "On the Generation of One-Time Keys in DL Signature Schemes," Presentation at IEEE P1363 Working Group meeting.

[9] Bogdanov, A., Guo , S., Masny, D., Richelson, R., and Rosen, A. (2016) "On the Hardness of Learning with Rounding over Small Modulus," in E. Kushilevitz and T. Malkin, TCC 2016-A, Springer LNCS 9562, 209–224.

[10] Boneh, D., and Shparlinski, I.E. (2001) "On the Unpredictability of Bits of the Elliptic Curve Diffie–Hellman Scheme," in J. Kilian (ed.) CRYPTO 2001, Springer LNCS 2139, 201–212.

[11] Boneh, D., and Venkatesan, R. (1996) "Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes," in N. Koblitz (ed.), CRYPTO 1996, Springer LNCS 1109, 129–142.

[12] Boneh, D., and Venkatesan, R. (1997) "Rounding in Lattices and its Cryptographic Applications," in Saks, M.E. (ed.) SODA 1997, ACM/SIAM, 675–681.

[13] De Mulder, E., Hutter, M., Marson, M.E., and Pearson, P. (2013) "Using Bleichenbacher's Solution to the Hidden Number Problem to Attack Nonce Leaks in 384-Bit ECDSA," in G. Bertoni and J.S. Coron (eds.), CHES 2013, Springer LNCS 8086, 435–452.

[14] Duc, A., and Jetchev, D. (2012) "Hardness of Computing Individual Bits for One-Way Functions on Elliptic Curves," in R. Safavi-Naini and R. Canetti (eds.), CRYPTO 2012, Springer LNCS 7417, 832–849.

[15] Fazio, N., Gennaro, R., Perera I.M., and Skeith, W.E. III (2013) "Hard-Core Predicates for a Diffie-Hellman Problem over Finite Fields," in R. Canetti and J. A. Garay (eds.), CRYPTO 2013, Springer LNCS 8043, 148–165.

[16] Galbraith, S.D., and Shani, B. (2015) "The Multivariate Hidden Number Problem," in Information Theoretic Security, 250–268.

[17] Gilbert, A.C., Indyk, P., Iwen, M., and Schmidt, L. (2014) "Recent Developments in the Sparse Fourier Transform," in IEEE Signal Processing Magazine, **31** (5), 91–100.

[18] Goldreich, O., and Levin, L.A. (1989) "A Hard-Core Predicate for all One-Way Functions," in STOC 1989, 25–32.

[19] González Vasco, and M.I., Näslund, M. (2001) "A Survey of Hard Core Functions," in Proc. Workshop on Cryptography and Computational Number Theory, 227–255.

[20] González Vasco, M.I., Näslund, M., and Shparlinski, I.E. (2004) "New Results on the Hardness of Diffie-Hellman Bits," in F. Bao, R.H. Deng and J. Zhou (eds.), PKC 2004, Springer LNCS 2947 159–172.

[21] González Vasco, and Shparlinski, I.E. (2001) "On the Security of Diffie-Hellman Bits," in Proc. Workshop on Cryptography and Computational Number Theory 1999, 257–268.

[22] Green, B., and Konyagin, S. (2009) "On the Littlewood Problem Modulo a Prime," in Canad. J. Math., vol. 61(1), 141–164.

[23] Håstad, J., and Näslund, M. (2003) "The Security of all RSA and Discrete Log Bits," in Journal of the ACM, **51(2)**, 187–230.

[24] Kawachi, A., and Yamane, I. (2015) "A Fourier-Analytic Approach to List-Decoding for Sparse Random Linear Codes," in IEICE Transactions on Information and Systems, **98-D(3)**, 532–540.

[25] Kushilevitz, E., and Mansour, Y. (1991) "Learning Decision Trees Using the Fourier Sprectrum," in Koutsougeras, C., Vitter, J.S. (eds.) STOC 1991, ACM, 455–464.

[26] Konyagin, S.V., and Shkredov, I.D. (2015) "A Quantitative Version of the Beurling-Helson Theorem," in Functional Analysis and Its Applications, vol. 49(2), 110–121.

[27] Laity, J., and Shani, B. (2016) "On Sets of Large Fourier Transform: A New Approach - Theory and Applications," work in progress.

[28] Langlois, A., Peikert, C., Regev, O., and Stehlé, D. (2013) "Classical Hardness of Learning with Errors," in STOC 2013, 575–584.

[29] Lebedev, V. (2012) "Absolutely Convergent Fourier Series. An Improvement of the Beurling-Helson Theorem," in Functional Analysis and Its Applications, vol. 46(2), 121–132.

[30] Li, W.-C. W., Näslund, M., and Shparlinski, I.E. (2002) "Hidden Number Problem with the Trace and Bit Security of XTR and LUC," in M. Yung (ed.), CRYPTO 2002, Springer LNCS 2442, 433–448.

[31] Ling, S., Shparlinski, I.E., Steinfeld, R., and Wang, H. (2011) "On the Modular Inversion Hidden Number Problem," in Journal of Symbolic Computation, **47(4)**, 358–367.

[32] Mansour, Y. (1992) "Randomized Interpolation and Approximation of Sparse Polynomials," in Proceedings of the 19th International Colloquium on Automata, Languages and Programming, 261–272.

[33] Mansour, Y. (1994) "Learning Boolean functions via the Fourier transform," in Theoretical advances in neural computation and learning, 391–424.

[34] Mansour, Y. and Sahar, S. (2000) "Implementation Issues in the Fourier Transform Algorithm," in Machine Learning, vol. 40(1), 5–33.

[35] Micciancio, D., and Mol, P. (2011) "Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions," in P. Rogaway (ed.), CRYPTO 2011, Springer LNCS 6841, 465–484.

[36] Moreno, C.J., and Moreno, O. (1991) "Exponential sums and Goppa codes: I," in Proc. Amer. Math. Soc., **111**, 523–531.

[37] Morillo, P., and Ràfols, C. (2009) "The Security of All Bits Using List Decoding," in S. Jarecki and G. Tsudik (eds.), PKC 2009, Springer LNCS 5443, 15–33.

[38] Nguyen, P.Q., and Shparlinski, I.E. (2002) “The Insecurity of the Digital Signature Algorithm with Partially Known Nonces,” in Journal of Cryptology, **15** (3), 151–176.

[39] Nguyen, P.Q., and Stern, J. (2001) “The Two Faces of Lattices in Cryptology,” in J. H. Silverman (ed.), Cryptography and Lattices 2001, Springer LNCS 2146, 146–180.

[40] Regev, O. (2005) “On Lattices, Learning with Errors, Random Linear Codes, and Cryptography,” in STOC 2005, 84–93.

[41] Shor, P.W. (1994) “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer,” in FOCS 1994, 124–134.

[42] Shparlinski, I.E. (2001) “Sparse Polynomial Approximation in Finite Fields,” in STOC 2001, 209–215.

[43] Shparlinski, I.E. (2005) “Playing “Hide-and-Seek” with Numbers: The Hidden Number Problem, Lattices and Exponential Sums,” in Public-Key Cryptography, Proceedings of Symposia in Applied Mathematics, vol. 62, AMS, pp. 153–177.

[44] Shparlinski, I.E., and Winterhof, A. (2005) “A Hidden Number Problem in Small Subgroups,” in Mathematics of Computation, vol. 74, pp. 2073–2080.

[45] Shparlinski, I.E., and Winterhof, A. (2004) “A Nonuniform Algorithm for the Hidden Number Problem in Subgroups,” in F. Bao, R.H. Deng and J. Zhou (eds.), PKC 2004, Springer LNCS 2947, 416–424.

[46] Terras, A. (1999) “Fourier Analysis on Finite Groups and Applications,” in London Mathematical Society Student Texts (No. 43), Cambridge University Press. Cambridge.

[47] Verheul, E.R. (2000) “Certificates of Recoverability with Scalable Recovery Agent Security,” in H. Imai and Y. Zheng (eds.), PKC 2000, Springer LNCS 1751, 258–275.

[48] Wang, M., Zhan, T., and Zhang, H. (2016) “Bit Security of the CDH Problems over Finite Fields,” in O. Dunkelman and L. Keliher (eds.), SAC 2015, Springer LNCS 9566, 441–461.