Proving Hard-Core Predicates Using List Decoding

Adi Akavia * Shafi Goldwasser † Samuel Safra ‡

ABSTRACT

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

Our framework extends the list-decoding method of Goldreich and Levin for showing hard-core predicates. Namely, a predicate will correspond to some error correcting code, predicting a predicate will correspond to access to a corrupted codeword, and the task of inverting one-way functions will correspond to the task of list decoding a corrupted codeword.

A characteristic of the error correcting codes which emerge and are addressed by our framework, is that codewords can be approximated by a small number of heavy coefficients in their Fourier representation. Moreover, as long as corrupted words are close enough to legal codewords, they will share a heavy Fourier coefficient. We list decode such codes, by devising a learning algorithm applied to corrupted codewords for learning heavy Fourier coefficients.

For codes defined over $\{0,1\}^n$ domain, a learning algorithm by Kushilevitz and Mansour already exists. For codes defined over Z_N, which are the codes which emerge for predicates based on number theoretic one-way functions such as the RSA and Exponentiation modulo primes, we develop a new learning algorithm. This latter algorithm may be of independent interest outside the realm of hard-core predicates.

1. INTRODUCTION

Let f be a one-way function, namely a function which is easy to compute, but hard to invert on all but a negligible fraction of its inputs. We say that a Boolean predicate P is a *hard-core predicate for* f if $P(x)$ is easy to compute given x, but hard to guess with non-negligible advantage beyond 50% given only $f(x)$. The notion of hard-core predicates was introduced and investigated in [9, 4] and has since proven central to cryptography and pseudo-randomness.

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

Blum and Micali [4] were the first to show a hard-core predicate for a function widely conjectured to be one-way. Let p be a prime and g a generator for Z_p^*. The function $EXP_{p,g}: Z_{p-1} \to Z_p^*$, $EXP_{p,g}(x) = g^x \bmod p$ is easy to compute and as hard to invert as solving discrete logarithm mod p. Blum and Micali[4] define the predicate $BM_{p,g}(x) = 0$ if $0 \le x < \frac{p-1}{2}$ and 1 otherwise, and proved it is a hard-core predicate for $EXP_{p,g}$ if the discrete logarithm problem is intractable. In subsequent years, it was shown for other conjectured one-way functions f and other predicates P, that P is a hard-core predicate for f [9, 22, 3, 20, 10, 1, 11, 13, 5]. Most notably, for the RSA [18] function $RSA: Z_n^* \to Z_n^*$, $RSA(x) = x^e \bmod n$, the predicates $P_i(x) = i^{th}$ bit of x were shown hard-core, first for $i = 1, |n|$ [1] and recently for any $1 \le i \le |n|$ [11].

Goldreich and Levin [7] address the general question of whether every one-way function (OWF) has some hard-core predicate. They show that for any OWF $f: \{0,1\}^n \to \{0,1\}^*$ one can define another OWF $f': \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^* \times \{0,1\}^n$ by $f'(x,r) = \langle f(x), r\rangle$, so that the predicate $GL(x,r) = \sum_{i=1}^n x_i y_i$ is a hard-core predicate for f'.

The work of Goldreich-Levin, which explicitly addressed hard-core predicates for arbitrary one-way functions, by way of solution gave a polynomial time list-decoding algorithm for a well known error correcting code – the Hadamard code. It introduced an interesting connection between hard-core predicates and list decoding which, as pointed out by Impagliazzo and Sudan [12, 19, 21], could potentially lead to a general *list decoding methodology* [1] for proving hard-core predicates for one-way functions.

In this methodology, given a function f and predicate P, one would have to:

1. **Define a Code.** Identify an error-correcting

[1]Interestingly, this view was not spelled out in '89 when the work came out, but is rather a later perspective which emerged with the growing interest in list-decoding and its applications.

* Computer Science and Applied Mathematics of Weizmann Institute, ISRAEL. akavia@wisdom.weizmann.ac.il

† Computer Science and Applied Mathematics of Weizmann Institute and Department of EECS and CSAIL, MIT. Research supported in part by the Minerva Foundation. shafi@theory.lcs.mit.edu

‡ School of Mathematics and School of Computer Science, Tel Aviv University, ISRAEL. Research supported in part by the Fund for Basic Research administered by the Israel Academy of Sciences, and a Bikura grant. safra@post.tau.ac.il

Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)
0272-5428/03 $17.00 © 2003 **IEEE**

code $\mathcal{C}^P$ encoding distinct x's, such that given only $f(x)$ and the capability to compute $P(z)$ on input $f(z)$, one can query-access the codeword for x, C^P_x. In the case of [7] the code defined was the Hadamard code which is a natural as $GL(x,r)$ is precisely the r-th entry of the Hadamard encoding of string x.

2. List Decode. Show a polynomial-time list-decoding algorithm for the code $\mathcal{C}^P$, which works with query access to a corrupted codeword and tolerates a $< \frac{1}{2} - \varepsilon$ fraction of error. In the case of Hadamard code, [7] indeed provides such a list decoding algorithm.

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

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

This is no doubt an elegant methodology but is it a **useful** methodology for proving hard-core predicate results for natural f's and P's? At the very least, can we define appropriate codes and corresponding list decoding algorithms so as to employ this methodology for proving existing hard-core predicate results of [4] for the EXP function, and the results of of [1, 11] for the RSA function ?

These questions are the starting point for our work.

1.1 Our Work

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

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

The error correcting codes which emerge and are addressed by our framework, are those in which codewords can be approximated by considering only a small number of the heavy coefficients in their Fourier representation. Moreover, as long as corrupted words are close enough to legal codewords, they will share a heavy Fourier coefficient. To list decode, we will first devise a learning algorithm applied to corrupted codewords for finding their heavy Fourier coefficients, and then find all codewords for which these coefficients are heavy.

Let us now elaborate on the hard-core predicates and one-way functions for which we apply our framework, and give a more detailed description of our technique.

1.1.1 Segment Predicates

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

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

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

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

In contrast with the notion of segment predicates, we remark that in the past most all predicates investigated correspond in a fairly direct way with bits in the inverse of $f(x)$. An exception is the work of [17] showing that $P_i(x) = ith$ bit of $ax + b \bmod p$ for randomly chosen a, b, p are hard-core predicates for one-way functions f.

1.1.2 New Proofs of Old Results

It is easy to see that the hard-core predicates of [4, 1, 13] for candidate one-way functions $EXP, RSA, Rabin$ and ECL, are special cases of the segment predicate defined above. Thus, we re-prove in an entirely different and uniform manner, all the results of [4, 1, 13]

In contrast to previous proofs, the technical essence of the new proofs is to define appropriate codes and to list decode them. These two tasks are independent of the one-way function in question and depend only on the predicate. The only consideration given to the one-way function is for devising a way to access the corrupted codeword (step 3 in the methodology). A task which turns out to be very simple in all the cases considered. We stress that the proofs obtained here are completely different than the previous ones. In particular, the proofs do not require the use the binary gcd algorithm used in previous proofs of the hard-core predicates for RSA [2, 1, 11], nor the square root extraction over finite fields as in [4].

We present a new proof method for simultaneous security of many bits. For this purpose we generalize the notion of balanced hard-core predicates to unbalanced ones. To prove simultaneous bit security, we will show that any violation of simultaneous bit security implies a predictor for some unbalanced hard-core predicate. Using this method we show that a class of functions called *segment functions* – an extension of segment predicates, are simultaneously secure for $RSA, Rabin, EXP$, and ECL. In particular, this implies simultaneous security of the $O(\log \log N)$ most significant bits for those candidate OWFs [20, 15] .

Finally, the new framework applies to proving Goldreich-

Levin hard-core predicate in a natural manner where indeed the appropriate code is the Hadamard code.

For a partial summary of these results, see table 1.

1.1.3 Technique: List Decoding by Learning Algorithm

The basic approach we take in the list decoding algorithm we develop is to analyze the *Fourier representation* of codewords and corrupted codewords. In particular, for all the codes we use, most of the weight in the Fourier representation of every codeword is concentrated on a small set of characters. We refer to such codes as *concentrated codes*. Furthermore, we prove in the concentration and agreement lemma 1 that if w is close to a legal codeword C_x and $\mathcal{C}$ is a concentrated code, then there exists a character χ_α which has a heavy coefficient in both w and C_x.

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

Turning this into a polynomial-time list decoding algorithm for each code considered requires showing the existence of a learning algorithm which given access to an arbitrary function (the corrupted codeword) learns its heavy Fourier coefficients, as well as a recovery algorithm. Whereas learning and recovery algorithms exist for the Hadamard code, we develop them anew for the new codes of section 5. As it turns out only one recovery and learning algorithm suffice for the codes of section 5. The recovery algorithm is elementary. In contrast, the learning algorithm is interesting on his own right and constitutes another contribution of this paper.

1.1.4 Learning Heavy Fourier Coefficients for Functions Over Z_N

We design an efficient algorithm that learns the heavy Fourier coefficients of arbitrary functions **defined over** Z_N.

Given a threshold τ and query access to a function $\mathsf{g}\colon Z_N \to \mathbb{C}$, the algorithm returns a short list containing all the Fourier coefficients of g with weight at least τ. The algorithm extends the work of Mansour and Kushilevitz [14] which solves the same problem for $\mathsf{g}\colon \{0,1\}^n \to \{\pm 1\}$.

We go beyond the application to proving hard-core predicates in which the functions concerned are defined over Z_N and generalize our learning algorithm to work for function defined over general abelian groups. Specifically, let function $\mathsf{g}\colon \mathcal{D} \to \mathbb{C}$ where $\mathcal{D}$ is a finite abelian group with k generators $g_1, ..., g_k$ of known orders $N_1, ..., N_k$ respectively. We design a algorithm which learns all the heavy Fourier coefficients of g, w.p. at leat $1-\delta$; and its running time is polynomial in $\log|\mathcal{D}|, 1/\tau, \|\mathsf{g}\|_\infty = \max_x |\mathsf{g}(x)|, \|\mathsf{g}\|_2$ and $\ln(1/\delta)$.

1.2 Related Work

Hastad and Naslund[11] showed that the ith bit of x (in its binary representation) is a hard-core predicate for the RSA function. We note that this is different than our result showing that the ith partition bit is a hard-core predicate for the RSA function. It is interesting to study further whether the same techniques can be applied to obtain both sets of results.

Fourier analysis has been looked at previously in the context of hard-core predicate in the work of Goldmann *et al* [6].

The literature of hard-core predicate is quite vast and many techniques have been employed throughout the years, which we cannot elaborate on here. The technique of Kaliski [13] for proving hard-core predicate for the discrete logarithm problem in general groups might be another interesting avenue to explore in the context of list decoding for discrete log based functions; it also does not use square root extraction of [4] as well.

1.3 Roadmap

In section 2 we provide preliminaries on Fourier analysis, and the necessary definitions for learning, binary codes, and one-way functions. In section 3, we present our algorithm for list decoding via learning, and prove its correctness based on the concentration and agreement lemma. In section 4, we formally describe the conditions for proving hard-core predicates via list decoding approach. In section 5 we focus on showing hard-core predicates for number theoretic functions. In section 6 we use our techniques to prove simultaneous bit security. In section 7 we describe our result on learning Fourier coefficients of functions over Z_N. Due to lack of space many proofs are omitted and are included in the full version.

2. PRELIMINARIES

2.1 Groups and Fourier Transform

Let $(\mathcal{D}, \cdot)$ denote a abelian group, where $\mathcal{D}$ is the set of group-elements and $\cdot$ is the group-operation. Consider now the set of all functions from $\mathcal{D}$ to the complex-numbers

$$\mathcal{V}_{\mathcal{D}\to\mathbb{C}} \stackrel{def}{=} \{\mathsf{g}\colon \mathcal{D} \to \mathbb{C}\}$$

This forms a vector space of dimension $|\mathcal{D}|$ over the complex-numbers. The expectation inner-product for this vector space is $\langle \mathsf{g}, \mathsf{h}\rangle \stackrel{def}{=} \mathbb{E}_{x\in\mathcal{D}}[\mathsf{g}(x)\cdot\overline{\mathsf{h}(x)}]$ (where $\overline{z}$ denotes the complex conjugate of z). The l_2-norm is defined, as standard, by, $\|\mathsf{g}\|_2^2 = \langle \mathsf{g}, \mathsf{g}\rangle$.

The natural basis for this vector space consists of all functions e_x for every x, where $e_x(x) = 1$ and $e_x(y) = 0$ for any $y \neq x$. An alternative basis for this vector-space is the Fourier basis, consisting of all multiplicative functions.

A *character* $\chi_\alpha\colon \mathcal{D} \to \mathbb{C}$ is a multiplicative function, namely,

$$\forall x, y \in \mathcal{D},\ \chi_\alpha(x\cdot y) = \chi_\alpha(x)\cdot\chi_\alpha(y)$$

These characters are orthogonal, each of l_2-norm 1, and there are $|\mathcal{D}|$ of them, hence they form an orthonormal basis for $\mathcal{V}_{\mathcal{D}\to\mathbb{C}}$, referred to as *the Fourier basis.*

DEFINITION 1 (FOURIER REPRESENTATION). *The* Fourier representation *of a function in $\mathcal{V}_{\mathcal{D}\to\mathbb{C}}$ is thus the vector of its projection on each of the characters, namely,*

$$\mathsf{g} = \sum_{\alpha\in\mathcal{D}} \widehat{\mathsf{g}}(\alpha)\chi_\alpha$$

where we denote

$$\widehat{\mathsf{g}}(\alpha) \stackrel{def}{=} \langle \mathsf{g}, \chi_\alpha\rangle.$$

Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)
0272-5428/03 $17.00 © 2003 **IEEE**

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

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

The coefficient $\widehat{\mathsf{g}}(\alpha)$ is called the α Fourier coefficient *of* g *and $|\widehat{\mathsf{g}}(\alpha)|^2$ is its* weight *(where for any complex number $z = a+ib$, $|z|^2 = a^2+b^2$).*

We normally utilize a one-to-one mapping between the elements of $\mathcal{D}$ and the characters, and thus index each character χ_α by an element $\alpha \in \mathcal{D}$.

DEFINITION 2 (RESTRICTION). *Given $\mathsf{g}\colon \mathcal{D} \to \mathbb{C}$ and a set of characters Γ. The* restriction of g to Γ *is the function $\mathsf{g}_{|\Gamma}\colon \mathcal{D} \to \mathbb{C}$ defined by*

$$\mathsf{g}_{|\Gamma} = \sum_{\chi_\alpha \in \Gamma} \widehat{\mathsf{g}}(\alpha)\chi_\alpha$$

We refer to g as *Fourier concentrated* if we can approximate it to within any ε by its restriction to a small, namely, of size $poly(\log(|\mathcal{D}|)/\varepsilon)$, set of characters:

DEFINITION 3 (FOURIER CONCENTRATION). *A function $\mathsf{g}\colon \mathcal{D} \to \mathbb{C}$ is* Fourier concentrated *if for every $\varepsilon > 0$ there exists a set Γ consisting of $poly(\log(|\mathcal{D}|)/\varepsilon)$ characters, so that the l_2-norm of* g *outside Γ ($\mathsf{g}_{|\bar{\Gamma}}$) is small*

$$\|\mathsf{g} - \mathsf{g}_{|\Gamma}\|_2 \le \varepsilon$$

2.2 Fourier Transform Over Z_N

The characters over the additive group Z_N are as follows.

DEFINITION 4 (CHARACTERS OVER Z_N). *For each $\alpha \in Z_N$, the α-*character *$\chi_\alpha\colon Z_N \to \mathbb{C}$ is defined by*

$$\chi_\alpha(x) \stackrel{def}{=} \omega_N^{\alpha x}$$

where $\omega_N = e^{i\frac{2\pi}{N}}$ is the primitive root of unity of order N.

The following proposition gives bounds on the expected value of a character χ_α when evaluated on an interval $\{0,...,l-1\}$. For $\alpha \in N$, denote $\mathsf{abs}(\alpha) \stackrel{def}{=} \min\{\alpha, N-\alpha\}$.

PROPOSITION 1. *Denote $S_l(\alpha) = \mathbb{E}_{y=0,...,l-1}[\omega^{\alpha y}]$*

- *Upper bound: $|S_l(\alpha)|^2 < \frac{(N/l)^2}{\mathsf{abs}(\alpha)^2}$*
- *Lower bound: If $\mathsf{abs}(\alpha) \le \frac{N}{2l}$, then $|S_l(\alpha)|^2 > \frac{1}{6}$.*

Proof. For ease of notation assume w.l.o.g $\alpha = \mathsf{abs}(\alpha)$.

Since $\sum_{y=0}^{l-1}\omega^{\alpha y}$ is a geometric sum,

$$S_l(\alpha) = \frac{1}{l}\frac{\omega^{\alpha l}-1}{\omega^\alpha - 1}$$

Noting that $\forall b, \left|\omega^b - 1\right|^2 = 2(1-\cos(\frac{2\pi}{N}b))$, and utilizing Taylor approximation, namely, $1-\frac{\theta^2}{2!} \le \cos(\theta) \le 1-\frac{\theta^2}{2!}+\frac{\theta^4}{4!}$ (for $|\theta| \le \pi$), we get

$$|S_l(\alpha)|^2 = \frac{1}{l^2}\frac{1-\cos(\frac{2\pi}{N}\alpha l)}{1-\cos(\frac{2\pi}{N}\alpha)} \le \frac{1}{l^2}\frac{2}{\frac{(\frac{2\pi}{N}\alpha)^2}{2!} - \frac{(\frac{2\pi}{N}\alpha)^4}{4!}} < \frac{(N/l)^2}{\alpha^2}$$

Now, if $\alpha \le \frac{N}{2l}$, then

$$|S_l(\alpha)|^2 \ge \frac{1}{l^2}\frac{\frac{(\frac{2\pi}{N}\alpha t)^2}{2!} - \frac{(\frac{2\pi}{N}\alpha t)^4}{4!}}{\frac{(\frac{2\pi}{N}\alpha)^2}{2!}} > \frac{1}{6}$$

□

2.3 Learnable Domain

A concentrated function $\mathsf{g}\colon \mathcal{D} \to \mathbb{C}$ can be approximated by a restriction to its heavy coefficients. We are therefore interested in learning those heavy coefficients, when given only a query access to g.

Let $\mathsf{Heavy}_\tau(\mathsf{g})$ be the set of all characters χ_α of weight at least τ

$$\mathsf{Heavy}_\tau(\mathsf{g}) \stackrel{def}{=} \{\chi_\alpha \mid |\widehat{\mathsf{g}}(\alpha)|^2 \ge \tau\}$$

Note that for functions whose l_2-norm is at most 1 (as are all functions discussed herein), $|\mathsf{Heavy}_\tau(\mathsf{g})| \le 1/\tau$, since by Parseval identity $\|\mathsf{g}\|_2^2 = \sum_\alpha |\widehat{\mathsf{g}}(\alpha)|^2$. In particular, for Boolean functions $\mathsf{g}\colon \mathcal{D} \to \{\pm 1\}$, $\|\mathsf{g}\|_2^2 = 1$.

DEFINITION 5 (LEARNABLE DOMAIN). *We say that $\mathcal{D}$ is* a learnable domain, *if there exists a learning algorithm that, given query access to a function $\mathsf{g}\colon \mathcal{D} \to \mathbb{C}$, and inputs τ, δ and 1^k (where $k = \log(|\mathcal{D}|)$ denotes the size of $x \in \mathcal{D}$), outputs a list of characters of length polynomial in $k, 1/\tau$ that contains $\mathsf{Heavy}_\tau(\mathsf{g})$, w.p. $1-\delta$; and the running time of the algorithm is polynomial in $k, 1/\tau, \log(\frac{1}{\delta}), \|\mathsf{g}\|_2, \|\mathsf{g}\|_\infty$.*

2.4 Binary Codes

An important notion utilized herein is that of a code, in particular, a binary-code. A binary-code is a subset of $\{\pm 1\}^*$, and concentrating on vectors of length m it is a set of *codewords* $\mathcal{C} \subseteq \{\pm 1\}^m$. When $m = |\mathcal{D}|$, we identify each function $\mathsf{g}\colon \mathcal{D} \to \{\pm 1\}$ with a vector in $\{\pm 1\}^m$, and think of $\mathcal{C}$ as a subset of

$$\mathcal{V}_{\mathcal{D}\to\{\pm 1\}} \stackrel{def}{=} \{\mathsf{g}\colon \mathcal{D} \to \{\pm 1\}\}$$

The codes discussed herein would be meant to encode elements of $\mathcal{D}$, thus there is a one-to-one mapping between

codewords and elements of $\mathcal{D}$:

$$\mathcal{C} = \{C_x\}_{x \in \mathcal{D}}$$

DEFINITION 6 (NORMALIZED HAMMING DISTANCE). *The* normalized Hamming distance *between two Boolean functions* $\mathsf{g}, \mathsf{h}: \mathcal{D} \to \{\pm 1\}$ *is*

$$\Delta(\mathsf{g}, \mathsf{h}) = \Pr_{x \in_R \mathcal{D}}[\mathsf{g}(x) \neq \mathsf{h}(x)]$$

For any Boolean function $\mathsf{g}: \mathcal{D} \to \{\pm 1\}$ denote

$$\mathsf{maj}_{\mathsf{g}} = \max_{b \in \{\pm 1\}} \Pr_{x \in_R \mathcal{D}}[\mathsf{g}(x) = b]$$

$$\mathsf{minor}_{\mathsf{g}} = 1 - \mathsf{maj}_{\mathsf{g}}$$

DEFINITION 7 (LIST DECODABLE). *We say that a code* $\mathcal{C} = \{C_x: \mathcal{D} \to \{\pm 1\}\}_{x \in \mathcal{D}}$ *is list decodable, if it has a list decoding algorithm. Namely, a PPT algorithm that, given access to a corrupted codeword* $w: \mathcal{D} \to \{\pm 1\}$ *and inputs* $\varepsilon, \delta, 1^k$ *(where* $k = \log|\mathcal{D}|$ *denotes the size of* $x \in \mathcal{D}$*), returns a list* $L \supseteq \{x \mid \Delta(C_x, w) \leq \mathsf{minor}_{C_x} - \varepsilon\}$ *w.p. at least* $1 - \delta$.

2.5 One-Way Functions and Hard-Core Predicates

We are interested in Boolean predicates. From this point on, we switch to work with range $\{\pm 1\}$ (rather than $\{0,1\}$).

We say that $\nu(\cdot)$ is *negligible* if for every constant $c \geq 0$ there exists an integer k_c such that $\nu(k) < k^{-c}$ for all $k \geq k_c$. Another way to think of it is $\nu(k) = k^{-\omega(1)}$.

We say that $\rho(\cdot)$ is *non-negligible* if there exists a constant $c \geq 0$ and an integer k_c such that $\rho(k) > k^{-c}$ for all $k \geq k_c$.

There are two equivalent definitions of (strong) one-way functions one may work with.

The first is a *single* function defined over an *infinite* domain which is asymptotically hard to invert with high probability (where the probability is taken over all strings of the same length).

The second definition is suitable for number theoretic OWFs candidates, and therefore, we use this formulation in this work.

DEFINITION 8 (OWF). *We say that* $\mathcal{F} = \{\mathsf{f}_i: \mathcal{D}_i \to R_i\}_{i \in I}$ *is a* collection of one-way functions (OWFs) *(where* I *is an infinite set of indices,* $\mathcal{D}_i$ *and* R_i *are finite), if (1) one can efficiently sample* $i \in I \cap \{0,1\}^k$ *(2)* $\forall i \in I$*, one can efficiently sample* $x \in \mathcal{D}_i$ *(3)* $\forall i \in I, x \in \mathcal{D}_i$*, one can efficiently compute* $\mathsf{f}_i(x)$ *and (4)* $\forall i \in I$*,* f_i *is (strongly) hard to invert, namely, for every PPT algorithm* A *there exists a negligible function* ν_A *such that*

$$\Pr[\mathsf{f}_i(z) = y: y = \mathsf{f}_i(x), z = A(i, y)] < \nu_A(k)$$

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

DEFINITION 9 (HARD-CORE PREDICATES). *Let* $\mathcal{P}$ *be a collection of Boolean predicates. We say that* $\mathcal{P}$ *is hard-core for a collection of one-way functions* $\mathcal{F}$ *if it is hard to guess the value of* $P_i(x)$ *from* $\mathsf{f}_i(x)$ *with any non-negligible advantage over a random guess, namely,* $\forall$ *PPT algorithm* B*, there exists a negligible function* ν_B*, s.t.*

$$\Pr[B(i, \mathsf{f}_i(x)) = P_i(x)] < \mathsf{maj}_{P_i} + \nu_B(k).$$

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

When $\mathcal{P}$ consists of *balanced predicates*[2], the condition is $\forall$ PPT algorithm B, $\exists$ a negligible function ν_B, s.t.

$$\Pr[B(i, \mathsf{f}_i(x)) = P_i(x)] < \frac{1}{2} + \nu(k).$$

DEFINITION 10 (PREDICTS). *We say that an algorithm* B predicts P_i *from* f_i *if* $\exists$ *a non-negligible function* ρ *s.t.*

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

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

In what follows we fix i and show an efficient reduction of inverting f_i with non-negligible probability to predicting $P_i(x)$ given $\mathsf{f}_i(x)$. This will suffice for showing that $\mathcal{P}$ is a hard-core predicate for $\mathcal{F}$.

For a full definition of one-way collection of functions and hard-core predicates for collections see [8].

3. LIST DECODING VIA LEARNING

In this section we present a list decoding algorithm for codes $\mathcal{C} = \{C_x: \mathcal{D} \to \{\pm 1\}\}_{x \in \mathcal{D}}$. The algorithm requires $\mathcal{D}$ to be a learnable domain, and $\mathcal{C}$ to be concentrated and recoverable (see definitions below).

DEFINITION 11 (CONCENTRATION). *We say that* $\mathcal{C}$ *is* concentrated *if each of its codewords* C_x *is a Fourier concentrated function.*

DEFINITION 12 (RECOVERY). *We say that* $\mathcal{C}$ *is* recoverable, *if it has a* recovery algorithm, *namely, a polynomial time algorithm that, given a character* χ_α *(for* $\alpha \neq 0$*), a threshold* τ*, and* 1^k *where* k *is the size of* x*'s in* $\mathcal{D}$*, returns a list* L_α *containing*

$$\{x \in \mathcal{D} \mid \mathsf{Heavy}_\tau(C_x) \ni \chi_\alpha\}$$

Given an input w we seek all codewords C_x that are close to w. The following lemma implies that when $\mathcal{C}$ is a concentrated code, if w is close to C_x, then w and C_x have a common heavy Fourier coefficient.

LEMMA 1 (CONCENTRATION AND AGREEMENT). *Let* $\mathsf{f}, \mathsf{g}: \mathcal{D} \to \mathbb{C}$ *where* $\|\mathsf{f}\|_2, \|\mathsf{g}\|_2 \leq 1$*, and such that* f *is a concentrated function and* $\langle \mathsf{f}, \mathsf{g} \rangle > \varepsilon + \left|\widehat{\mathsf{f}}(0)\widehat{\mathsf{g}}(0)\right|$ *for some* $\varepsilon > 0$*. Then, there exists an (explicit) threshold* τ *which is polynomial in* $\varepsilon, 1/k$ *(where* $k = \log(|\mathcal{D}|)$ *is the length of inputs* $x \in \mathcal{D}$*) such that*

$$\exists \alpha \neq 0, \chi_\alpha \in \mathsf{Heavy}_\tau(\mathsf{f}) \cap \mathsf{Heavy}_\tau(\mathsf{g})$$

Proof. Let Γ be a set of characters on which f is concentrated to within $o(\varepsilon)$ - namely, so that, if we denote by $\bar{\Gamma}$ its complement, it is the case that $\varepsilon' = \|\mathsf{f}_{|\bar{\Gamma}}\|_2 \leq o(\varepsilon)$. Note that, by Cauchy-Schwartz

$$\langle \mathsf{f}_{|\bar{\Gamma}}, \mathsf{g}_{|\bar{\Gamma}} \rangle^2 \leq \|\mathsf{f}_{|\bar{\Gamma}}\|_2^2 \cdot \|\mathsf{g}_{|\bar{\Gamma}}\|_2^2 \leq \varepsilon'^2 \cdot 1 = \varepsilon'^2$$

[2] $\mathcal{P}$ is a collection of balanced predicates if there exists a negligible function ν, s.t.

$$\left|\Pr_x[P_i(x) = 1] - \Pr_x[P_i(x) = -1]\right| < \nu(k)$$

and since the Fourier basis is orthonormal

$$\sum_{\chi_\alpha \in \Gamma} \widehat{\mathsf{f}}(\alpha)\widehat{\mathsf{g}}(\alpha) = \langle \mathsf{f}_{|\Gamma}, \mathsf{g}_{|\Gamma} \rangle \geq \langle \mathsf{f}, \mathsf{g} \rangle - \left|\langle \mathsf{f}_{|\bar{\Gamma}}, \mathsf{g}_{|\bar{\Gamma}} \rangle\right| \geq \varepsilon + \left|\widehat{\mathsf{f}}(0)\widehat{\mathsf{g}}(0)\right| - \varepsilon'$$

which implies there exists a $\alpha \neq 0$ s.t. $\chi_\alpha \in \Gamma$ and $\left|\widehat{\mathsf{f}}(\alpha)\widehat{\mathsf{g}}(\alpha)\right| \geq \frac{\varepsilon - \varepsilon'}{|\Gamma|}$. Now, as both $\left|\widehat{\mathsf{f}}(\alpha)\right|, |\widehat{\mathsf{g}}(\alpha)| \leq 1$, it must be that both $\left|\widehat{\mathsf{f}}(\alpha)\right|, |\widehat{\mathsf{g}}(\alpha)| \geq \frac{\varepsilon - \varepsilon'}{|\Gamma|} = \tau$ □

Now, we present the list decoding algorithm.

THEOREM 1 (LIST DECODING VIA LEARNING). *Let $\mathcal{D}$ be a learnable domain, and let $\mathcal{C} = \{C_x : \mathcal{D} \to \{\pm 1\}\}$ be a concentrated and recoverable code, then $\mathcal{C}$ is list decodable.*

Proof. Given $\varepsilon, \delta, 1^k$ and a corrupted codeword w, consider a codeword C_x such that $\Delta(C_x, w) \leq \mathsf{minor}_{C_x} - \varepsilon$.

Replacing f, g in the the concentration and agreement lemma above with C_x, w respectively, and noting that

$$\Delta(C_x, w) \leq \mathsf{minor}_{C_x} - \varepsilon \text{ iff } \langle C_x, w \rangle \geq \left|\mathop{\mathbb{E}}_j[C_x(j)]\right| + \varepsilon$$

we conclude that there exist a threshold τ which is polynomial in $\varepsilon, 1/k$, and a character χ_α with $\alpha \neq 0$, so that

$$\chi_\alpha \in \mathsf{Heavy}_\tau(C_x) \cap \mathsf{Heavy}_\tau(w)$$

Since $\mathcal{D}$ is a learnable domain, we can in time polynomial in $k, \varepsilon^{-1}, \log(\frac{1}{\delta})$ find a list L' that contains $\mathsf{Heavy}_\tau(w)$, w.p. $1 - \delta$.

Let the output of the list decoding algorithm be the list

$$L \supseteq \left\{ x \in \mathcal{D} \mid \mathsf{Heavy}_\tau(C_x) \cap L' \neq \phi \right\}$$

obtained by applying the recovery algorithm with threshold τ on each character in L'.

Since $\mathsf{Heavy}_\tau(C_x) \ni \chi_\alpha$, it must be that $x \in L$ with high probability. Since τ is polynomial in $\varepsilon, 1/k$, then the length of the list L and the running time of the algorithm are polynomial in $k, 1/\varepsilon, \log(\frac{1}{\delta})$. □

4. HARD-CORE PREDICATES VIA LIST DECODING

Throughout this section $\mathcal{F} = \{\mathsf{f}_i : \mathcal{D}_i \to R_i\}_{i \in I}$ denotes a collection of OWFs, $\mathcal{P} = \{P_i : \mathcal{D}_i \to \{\pm 1\}\}_{i \in I}$ denotes a collection of predicates, and $\mathcal{C}^{\mathcal{P}} = \left\{\mathcal{C}^{P_i}\right\}_{i \in I}$ denotes a collection of codes, where $\mathcal{C}^{P_i} = \left\{C_x^{P_i} : \mathcal{D}_i \to \{\pm 1\}\right\}$ is a code with codewords $C_x^{P_i}$ corresponding to a non-negligible fraction of the $x \in \mathcal{D}_i$.

DEFINITION 13 (ACCESSIBLE). *Let $\mathcal{P}$ be a collection of predicates. We say that $\mathcal{C}^{\mathcal{P}}$* is accessible w.r. to F, *if there exists a PPT access algorithm A, s.t. $\forall i \in I \cap \{0,1\}^k$, $\mathcal{C}^{P_i}$* is accessible w.r. to f_i, *namely,*

1. **Code access**: *$\forall x, j \in \mathcal{D}_i$, $A(i, \mathsf{f}_i(x), j)$ returns $\mathsf{f}_i(x')$ s.t. $C_x^{P_i}(j) = P_i(x')$.*
2. **Well spread**: *For uniformly distributed $C_x^{P_i} \in \mathcal{C}^{P_i}$ and $j \in \mathcal{D}_i$, the distribution of x' satisfying $\mathsf{f}_i(x') = A(i, \mathsf{f}_i(x), j)$ is statistically close[3] to the uniform distribution on $\mathcal{D}_i$.*
3. **Bias preserving**: *For every codeword $C_x^{P_i} \in \mathcal{C}^{P_i}$, $\left|\Pr_j[C_x^{P_i}(j) = 1] - \Pr_z[P_i(z) = 1]\right| \leq \nu(k)$, where ν is a negligible function.*

[3]Namely, $\frac{1}{2}\sum_{c \in \mathcal{D}_i} |\Pr_{x'}[x' = c] - \Pr_z[z = c]| < \nu(k)$ for a negligible function ν.

For ease of notations, we fix some $i \in I \cap \{0,1\}^k$ and drop the indices. We now show that if $\mathcal{C}^P$ is accessible w.r. to f, then an algorithm B that predicts P implies access to corrupted codewords.

LEMMA 2. *Let $P : \mathcal{D} \to \{\pm 1\}$ be a balanced predicate. Assume $\mathcal{C}^P$ is accessible w.r. to f. Assume we are given a PPT algorithm B that predicts P from f. Then, for a non-negligible fraction of the codewords $C_x^P \in \mathcal{C}^P$, given $\mathsf{f}(x)$, we have access to a corrupted codeword w_x satisfying*

$$\Delta(w_x, C_x^P) \leq \frac{1}{2} - \rho(k)$$

where ρ is a non-negligible function.

Proof. As $\mathcal{C}^P$ is accessible w.r. to f, there exists an access algorithm A as in definition 13.

Give $\mathsf{f}(x)$, define[4] w_x by

$$w_x(j) = B(A(\mathsf{f}(x), j))$$

Let $a_{x,j} \in \mathcal{D}$ satisfy $\mathsf{f}(a_{x,j}) = A(\mathsf{f}(x), j)$. Since the code is well spread, and B predicts P with some non-negligible advantage ρ',

$$\Pr[B(\mathsf{f}(a_{x,j})) = P(a_{x,j})] \geq \frac{1}{2} + \rho'(k) - \nu(k)$$

where the probability is taken over random coin tosses of B and over random choices of $C_x^P \in \mathcal{C}^P$ and $j \in \mathcal{D}$.

Let $2\rho(k) = \rho'(k) - \nu(k)$. Thinking of $a_{x,j}$ as first choosing $C_x \in S$, and then $j \in \mathcal{D}$, by a counting argument, $\exists S' \subseteq \mathcal{C}^P$ of at least $\rho(k)$ fraction of the codewords s.t.

$$\forall C_x^P \in S',\ \Pr[B(\mathsf{f}(a_{x,j})) = P(a_{x,j})] \geq \frac{1}{2} + \rho(k)$$

where the probability is taken over random coin tosses of B and over random choices of $j \in \mathcal{D}$. Namely, $\forall C_x^P \in S'$, $\Delta(w_x, C_x^P) \geq \frac{1}{2} - \rho(k)$. □

For unbalanced predicates we prove a similar lemma.

LEMMA 3. *Let $P : \mathcal{D} \to \{\pm 1\}$ be a predicate. Assume $\mathcal{C}^P$ is accessible w.r. to f. Assume we are given a PPT algorithm B that predicts P from f. Then, for a non-negligible fraction of the codewords $C_x^P \in \mathcal{C}^P$, given $\mathsf{f}(x)$, we have access to a corrupted codeword w_x satisfying*

$$\Delta(w_x, C_x^P) \leq \mathsf{minor}_{C_x^P} - \rho(k)$$

for ρ a non-negligible function.

EXAMPLE 1 (BINARY HADAMARD CODE IS ACCESSIBLE). *Let $\mathcal{F}' = \{\mathsf{f}'_n : \{0,1\}^n \times \{0,1\}^n \to R_n\}$ denote a collection of OWFs, where $\mathsf{f}'_n(x,y) = \mathsf{f}_n(x).y$ is the concatenation of $\mathsf{f}_n(x)$ and y. Let $GL = \{GL_n : \{0,1\}^n \times \{0,1\}^n \to \{\pm 1\}\}$ be the collection*

[4]Although A, B are probabilistic algorithms, w.l.o.g we assume w_x is well-defined: Since our list decoding algorithm accesses w_x only polynomially many times, by taking $w_x(j)$ to be the majority value in polynomially many applications of A, B, we have only a negligible probability to of encountering different values for the same entry $w_x(j)$.

of predicates: $GL_n(x,r) = (-1)^{\langle x,r\rangle}$. *Let* $\mathcal{C}^{GL} = \{\mathcal{C}^{GL_n}\}$ *be the collection of* binary Hadamard codes*:*

$$\mathcal{C}^{GL_n} = \{C_{x,y}\colon \{0,1\}^n \to \{\pm 1\},\ C_{x,y}(j) = (-1)^{\langle x,j\rangle}\}$$

Then the algorithm $A(n, \mathsf{f}'_n(x,y),(j',j)) = \mathsf{f}_n(x).j$ *is an access algorithm for* $\mathcal{C}^{GL_n}$ *w.r to* f'_n*, and* $S = GL_n$ *is well spread, and as balanced as* GL_n*.*

THEOREM 2 (LIST DECODING APPROACH). *Assume a collection of codes* $\mathcal{C}^{\mathcal{P}} = \{\mathcal{C}^{P_i}\}_{i\in I}$*, s.t.* $\forall i \in I$*, (1)* $\mathcal{C}^{P_i}$ *is list decodable, and (2)* $\mathcal{C}^{P_i}$ *accessible w.r. to* f_i*. Then* $\mathcal{P}$ *is hard-core of* $\mathcal{F}$*.*

Proof. It suffices to show for a non-negligible fraction of the indices $i \in I$ a reduction of inverting f_i with non-negligible probability to predicting P_i from f_i. For ease of notation, in the rest of the proof we fix some $i \in I \cap \{0,1\}^k$ which satisfies items (2),(3) above, and drop the indices.

Assume an algorithm B that predicts P from f. Then, by lemma 3 above, there exists a non-negligible function ρ and a non-negligible fraction of the codewords $C_x^P \in \mathcal{C}^P$ s.t. we have random access to a corrupted codeword w_x satisfying $\Delta(w_x, C_x^P) \leq \mathsf{minor}_{C_x^P} - \rho(k)$.

We list-decode w_x to obtain a short list L containing x. Evaluating f on every candidate x' in L we output x' such that $\mathsf{f}(x') = \mathsf{f}(x)$ thus inverting $\mathsf{f}(x)$. □

REMARK 1. *In fact it suffices to have codes* $\mathcal{C}^{P_i}$ *satisfying items (2),(3) of the above theorem for non-negligible fraction of the indices* I*.*

REMARK 2. *Since our list decoding algorithm accesses the corrupted codeword* w_x *only polynomially many time, it cannot distinguish between two codewords which are within negligible distance from each other. Consequently, our proof of* $\mathcal{P}$ *being hard-core for* $\mathcal{F}$ *implies that* $\mathcal{P} = \{P'_i\}_{i\in I}$ *is also hard-core for* $\mathcal{F}$*, as long as each codeword* $C_x \in \mathcal{C}^{P'_i}$ *is within negligible distance from the codeword* $C_x \in \mathcal{C}^{P_i}$*.*

5. NUMBER THEORETIC HARD-CORE PREDICATES

Let Z_N be the ring of integers with addition and multiplication modulo N. In this section we prove that a broad family of predicates over Z_N, namely, *segment predicates*, is hard-core for the candidate one-way functions EXP, RSA, *Rabin* and ECL. The definition of segment predicates includes as a special case predicates previously shown hard-core [4, 1] as well as other predicates not previously known to be hard-core.

5.1 Segment Predicates

DEFINITION 14 (SEGMENT PREDICATE). *Let* $\mathcal{P} = \{P_N\colon Z_N \to \{\pm 1\}\}$ *be a collection of predicates that are non-negligibly far from constant, namely,* $\exists$ *non-negligible function* ρ *s.t.* $\mathsf{maj}_{P_N} \leq 1 - \rho(k)$ *where* $k = \log N$*.*

- *We say that* P_N *is a* basic t-segment predicate *if* $P_N(x+1) \neq P_N(x)$ *for at most* t x*'s in* Z_N*.*
- *We say that* P_N *is a* t-segment predicate *if there exist a basic* t*-segment predicate* P' *and* $a \in Z_N$ *which is co-prime to* N *s.t.* $\forall x \in Z_N, P_N(x) = P'(x/a)$*.*
- *If* $\forall N$*,* P_N *is a* $t(N)$*-segment predicate, where* $t(N)$ *is polynomial in* $\log N$*, we say that* $\mathcal{P}$ *is a collection of* segment predicates*.*

REMARK 3. *Requiring that* $\mathcal{P}$ *is non-negligibly far from constant is not essential. If it is close to constant then, trivially,* $P_i(x)$ *cannot be predicted with a* **non-negligible advantage** *over guessing its majority value, as the majority guess is already extremely good.*

The definition of segment predicates is quite general. It captures many of the previous predicates considered for RSA, *Rabin*, EXP and ECL as well as new predicates. In the following we illustrate the generality and ease of working with the segment predicate definition.

5.1.1 Examples of Segment Predicates

Most-significant bit is a segment predicate. Let $msb\colon Z_N \to \{\pm 1\}$ be defined by $msb(x) = 1$ if $x < N/2$, and -1 otherwise. This is a basic 2-segment predicate, since it changes value only twice.

Least-significant of RSA is a segment predicate. Let $lsb\colon \{0,...,N-1\} \to \{\pm 1\}$ be defined by $lsb(x) = 1$ iff x is even. When N is odd, $\forall x$, $lsb(x) = msb(x/2)$, thus as msb is a basic 2-segment predicate, lsb is a 2-segment predicate with $a = 2$. Consequently, lsb is a segment predicate for RSA, as well as for any other function over Z_N, where N is odd; but it is not a segment predicate for functions over an even domain such as EXP (where the domain Z_{p-1} is even, since p is prime).

Partition bits of RSA are segment predicates. We define the i-th partition bit, $b_i\colon Z_N \to \{\pm 1\}$, to give alternating values on a partition of Z_N to 2^i intervals, namely, $b_i(x) = msb(x2^i)$. This is a natural generalization of the most-significant bit (corresponding to $i = 0$), in which Z_N is partitioned into two halves. Again as $msb(x)$ is a basic 2-segment predicate, b_i is a 2-segment predicate with $a = 2^{-i}$ for RSA, as well as for any other function over Z_N, where N is odd.

Example of a new basic segment predicate. In general, we can define a new basic segment predicate by choosing any partition of Z_N to polynomially many intervals, and giving arbitrary answer (in ± 1) for each interval. For example, a segment predicate might give 1 on the first 10% of the inputs and between the middle $50-90\%$, and -1 otherwise.

Example of a new segment predicate. From any basic segment predicate, and any a co-prime to N one may define a (general) segment predicate. For example, in the EXP case, though the lsb is not a segment predicate, many others are. For instance, consider a trinary partition of the msb defined by $TriLsb(x) \stackrel{def}{=} msb(x/3)$. Under the standard assumption that $p - 1 = 2q$, where q is prime, this is a segment predicate.

5.2 Defining a Code

DEFINITION 15 (MULTIPLICATION CODE). *For each predicate* $P_N\colon Z_N \to \{\pm 1\}$*, we define the* multiplication code $\mathcal{C}^{P_N} = \{C_x\colon Z_N \to \{\pm 1\}\}_{x\in Z_N^*}$ *by*

$$C_x(j) = P_N(j \cdot x \bmod N)$$

For a collection of predicates $\mathcal{P} = \{P_N\colon Z_N \to \{\pm 1\}\}$*, denote* $\mathcal{C}^{\mathcal{P}} = \{\mathcal{C}^{P_N}\}$*.*

Note that $\mathcal{C}^{P_N}$ consists of codewords C_x for a non-negligible fraction of the $x \in Z_N$ (since Z_N^* is a non-negligible fraction of Z_N).

5.3 List Decoding

The additive group of Z_N is shown to be a learnable domain in theorem 6 in section 7.

In the next two lemmata we show that when P_N is a segment predicate, the multiplication code $\mathcal{C}^{P_N}$ is concentrated and recoverable. In the following, for any function g, we say that g *is concentrated within* ε *on* Γ, whenever $\|g - g_{|\Gamma}\|_2^2 \leq \varepsilon$.

LEMMA 4 (CONCENTRATION). *Let $\mathcal{P} = \{P_N\}$ be a collection of segment predicates, then $\forall N$, C^{P_N} is concentrated.*

Proof. We first prove that any basic segment predicate is concentrated on small characters (namely those for which $\mathsf{abs}(\alpha) = \min\{\alpha, N-\alpha\}$ is small):

CLAIM 4.1. *Let $\varepsilon > 0$. For a basic t-segment predicate $P: Z_N \to \{\pm 1\}$, P is concentrated within ε on $\Gamma = \{\chi_\alpha | \mathsf{abs}(\alpha) \leq O(t^2/\varepsilon)\}$, i.e*

$$\|P_{|\{\chi_\alpha \mid \mathsf{abs}(\alpha) > O(t^2/\varepsilon)\}}\|_2^2 \leq \varepsilon$$

PROOF. Let us first examine the Fourier coefficients of a basic 2-segment predicate P, which gives 1 on a segment I and -1 otherwise:

$$\left|\widehat{P}(\alpha)\right| = \mathop{\mathbb{E}}_x[P(x)\chi_\alpha(x)] = \frac{1}{N}[\sum_{x\in I}\chi_\alpha(x) - \sum_{x\notin I}\chi_\alpha(x)]$$

By Proposition 1, $\left|\frac{1}{N}\sum_{y=0}^{l-1}\chi_\alpha(y)\right| < \frac{1}{\mathsf{abs}(\alpha)}$. Consequently, since $\widehat{P}(\alpha)$ is the difference of two sums $\sum\chi_\alpha(x)$, and each can be expressed as a difference of two such sums initiated at 0, then $\left|\widehat{P}(\alpha)\right| < O(1/\mathsf{abs}(\alpha))$.

Now, let us examine a basic t-segment predicate P. A basic t-segment predicate defines a partition of Z_N into t segment I_j, so that P is constant on each segment I_j. Thus we can express P as a sum, $P = t-1+\sum_{j=1}^{t} P_j$, of functions $P_j: Z_N \to \{\pm 1\}$ such that $P_j(x)$ is the constant $P(x)$ for $x \in I_j$ and -1 otherwise.

Note that each P_j is a basic 2-segment predicate, thus $\left|\widehat{P_j}(\alpha)\right| < O(1/\mathsf{abs}(\alpha))$. Therefore,

$$\left|\widehat{P}(\alpha)\right| = \left|\sum_{j=1}^{t}\widehat{P_j}(\alpha)\right| \leq O(t/\mathsf{abs}(\alpha))$$

Now, consider the sum of weights over all large characters χ_α:

$$\sum_{\mathsf{abs}(\alpha)>k}\left|\widehat{P}(\alpha)\right|^2 \leq O(t^2)\sum_{\mathsf{abs}(\alpha)>k}\frac{1}{\mathsf{abs}(\alpha)^2} < O(\frac{t^2}{k})$$

which implies that for $\varepsilon > 0$

$$\|P_{|\{\chi_\alpha \mid \mathsf{abs}(\alpha) > O(t^2/\varepsilon)\}}\|_2^2 \leq \varepsilon$$

■

From the above Claim, we can easily deduce that $\mathcal{C}^{P_N}$ is concentrated, by applying the following claim:

CLAIM 4.2. *For $\mathsf{f}, \mathsf{g}: Z_N \to \mathbb{C}$ such that $\mathsf{g}(y) = \mathsf{f}(y/b)$ where $b \in Z_N^*$, $\widehat{\mathsf{g}}(\alpha) = \widehat{\mathsf{f}}(\alpha b)$.*

PROOF. By definition $\widehat{\mathsf{g}}(\alpha) = \mathbb{E}_{y\in Z_p}[\mathsf{f}(y/b)\chi_\alpha(y)]$. Since $b \in Z_N^*$, $\{yb\}_{y\in Z_N} = Z_N$, thus $\widehat{\mathsf{g}}(\alpha) = \mathbb{E}_{yb, y\in Z_N}[\mathsf{f}(yb/b)\chi_\alpha(yb)]$. Now, as $\chi_\alpha(yb) = \chi_{\alpha b}(y)$, $\widehat{\mathsf{g}}(\alpha) = \mathbb{E}_y[\mathsf{f}(y)\chi_{\alpha b}(y)] = \widehat{\mathsf{f}}(\alpha b)$. ■

The code $\mathcal{C}^{P_N}$ is defined by $C_x(j) = P_N(jx)$, for P_N a general segment predicate. Since, $P_N(x) = P_N{}'(x/a)$ (where $P_N{}'$ is a basic t-segment predicate), $C_x(j) = P_N{}'(jx/a)$. Now, by the above Lemma, $P_N{}'$ is concentrated within ε on $\{\chi_\alpha \mid \mathsf{abs}(\alpha) \leq O(t^2/\varepsilon)\}$ and therefore, C_x is concentrated to within ε on

$$\Gamma = \{\chi_\beta \mid \beta = \alpha(x/a) \bmod N, \mathsf{abs}(\alpha) \leq O(t^2/\varepsilon)\}$$

Thus $\mathcal{C}^{P_N}$ is concentrated. □

LEMMA 5 (RECOVERY). *Let $\mathcal{P} = \{P_N\}$ be a collection of segment predicates, then $\forall N$, $\mathcal{C}^{P_N}$ is recoverable.*

Proof. Let P_N be a t-segment predicate. We show a recovery algorithm for $\mathcal{C}^{P_N}$ that runs in time polynomial in $t \log N$. Denote $P_N(y) = P_N{}'(y/a)$ for a basic segment predicate $P_N{}'$; then, $C_x(j) = P_N(jx) = P_N{}'(jx/a)$ implies (by Lemma 4) C_x is concentrated to within τ on

$$\Gamma = \{\chi_\beta \mid \beta = \alpha(x/a) \bmod N, \mathsf{abs}(\alpha) < O(t^2/\tau)\}$$

On input a character χ_β for $\beta \neq 0$ and a threshold parameter τ, let us describe an algorithm outputting a list containing $\{x \mid \mathsf{Heavy}_\tau(C_x) \ni \chi_\beta\}$.

Since C_x is concentrated to within τ on Γ, $\chi_\beta \in \mathsf{Heavy}_\tau(C_x)$ implies $\chi_\beta \in \Gamma$ and thus

$$\beta \equiv x(\alpha/a) \mod N$$

for $\mathsf{abs}(\alpha) \leq poly(\log N/\tau)$. Our algorithm outputs the union of the lists L_α, such that L_α contains all x so that $x \equiv \beta(\alpha/a)^{-1} \mod N$.

If α is co-prime to N, then so is α/a and there is a single solution to this equation, which can be efficiently computed by the Extended Euclid Algorithm. Otherwise, if α is not co-prime to N, then we can easily find $d = gcd(\alpha, N)$, compute

$$x \equiv \beta(\alpha/a)^{-1} \mod \frac{N}{d}$$

and return all

$$L_\alpha = \left\{x + i\cdot\frac{N}{d} \pmod N\right\}_{i=0,\ldots,d-1}$$

The union of the lists L_α (over all α such that $\mathsf{abs}(\alpha) \leq O(t^2/\tau)$) contains all x such that $\mathsf{Heavy}_\tau(C_x) \ni \chi_\beta$, and, since $gcd(\alpha, N)$ is small, the lengths of the lists and the time of constructing them are $poly(\log(N)/\tau)$. □

Combining Theorem 1 with the above lemmata, we conclude that for $\mathcal{P} = \{P_N\}$ a collection of segment predicates, for every N, the multiplication code $\mathcal{C}^{P_N}$ is list decodable.

5.4 Accessibility

We now show that that the multiplication code $\mathcal{C}^{P_N} = \{C_x\}_{x\in Z_N^*}$ is accessible w.r. to the well known candidate one-way functions of *RSA*, *Rabin*, *EXP* and *ECL*. As it turns out this task is simple for all the functions we considered. Due to lack of space, in this version we elaborate only on the *RSA* and *EXP* examples.

5.4.1 Accessibility w.r. to RSA

DEFINITION 16 (RSA). *Assuming hardness of inverting the RSA function yields the following collection of OWFs. Define* $RSA = \{RSA_{n,e}(x) = x^e \bmod n,\ RSA_{n,e}: Z_n^* \to Z_n^*\}_{\langle n,e\rangle \in I}$ *for* $I = \{\langle n,e\rangle,\ n = pq,\ |p| = |q|,\ p \text{ and } q \text{ are primes and } (e, \phi(n)) = 1\}$

One technical issue we need to address, when considering segment predicates in the context of *RSA* or *Rabin*, is that segment predicates were defined over the domain Z_N whereas $RSA, Rabin$ are defined over Z_N^*. To overcome this difficulty we extend the definition of segment predicates, by saying that $\mathcal{P}' = \{P'_N\} : Z_N^* \to \{\pm 1\}$ is a collection of segment predicates, if there exists a collection $\mathcal{P} = \{P_N : Z_N \to \{\pm 1\}\}$ of segment predicate, such that P'_N is the restriction of P_N to inputs from the domain Z_N^*. W.l.o.g assume $P_N(x) = 0$ for every $x \in Z_N \setminus Z_N^*$.

LEMMA 6 (*RSA*). *Let* $\mathcal{P} = \left\{ P_{N|Z_N^*} \;\middle|\; P_N : Z_N \to \{\pm 1\} \right\}$ *be a collection of segment predicates, then* $\mathcal{P}$ *is a hard-core predicate of RSA.*

Proof. Let the access algorithm A be

> **input:** $\langle N, e\rangle, RSA_{N,e}(x), j$
> **output:** If $j \in Z_N^*$, return $RSA_{N,e}(jx) = RSA_{N,e}(j) \cdot RSA_{N,e}(x) \bmod N$; else return 0.

For any fixed $x \in Z_N^*$, and uniformly distributed $j \in Z_N$, consider the distribution of x' satisfying $RSA_{N,e}(x') = A(i, RSA_{N,e}(x), j)$. The restriction of this distribution to Z_N^* is uniform; while its restriction to $Z_N \setminus Z_N^*$ gives only one value $x' = 0$. Still, this distribution is close to uniform, as w.l.o.g there is only a negligible fraction of inputs $j \notin Z_N^*$ (otherwise, RSA can be broken). Therefore, the code is well spread, and bias preserving. □

5.4.2 Accessibility w.r. to EXP

DEFINITION 17 (EXP). *Assuming hardness of solving the Discrete Log Problem yields the following collection of OWFs. Define* $EXP = \{EXP_{p,g}(x) = g^x \bmod p,\ EXPp, g: Z_{p-1} \to Z_p^*\}_{\langle p,g\rangle \in I}$ *for* $I = \{\langle p, g\rangle,\ p \text{ prime, } g \text{ generator for } Z_p^*\}$.

LEMMA 7 (EXP). *Let* $\mathcal{P} = \{P_{p,g} : Z_{p-1} \to \{\pm 1\}\}$ *be a collection of segment predicates, then* $\mathcal{P}$ *is a hard-core predicate of EXP.*

Proof. Let the access algorithm A be

> **input:** $\langle p, g\rangle, EXP_{p,g}(x), j$
> **output:** Return $EXP_{p,g}(xj) = EXP_{p,g}(x)^j \bmod N$.

For any fixed $x \in Z_{p-1}^*$, and uniformly distributed $j \in Z_{p-1}$, the distribution of x' satisfying $EXP_{p,g}(x') = A(i, EXP_{p,g}(x), j)$ is uniform. Therefore, the code is well spread, and bias preserving. □

REMARK 4. *So far, we invert* $EXP_{p,g}(x)$ *only for* x*'s which are co-prime to* $p-1$ *(as* $C^P = \{C_x\}_{x \in Z_{p-1}^*}$*). Nonetheless, by random self reducibility of EXP we can invert for any* x*: For* $r \in_R Z_{p-1}$, $EXP_{p,g}(x + r) = g^x \cdot g^r$ *is a generator of* Z_p^* *w.h.p. In this case, we can invert* $EXP_{p,g}(x + r)$ *to find* $x' = x + r$*, and return* $x = x' - r$.

REMARK 5. *Interestingly, lsb is a segment predicate over odd domains, and thus hard-core for RSA, while it is easy for EXP (where the domain is even). To see where the proof fails, consider the code* C^{lsb} *consisting of codewords* $C_x(j) = lsb(jx)$.

This code has no recovery algorithm with a succinct output, as its codewords correspond only to one out of two functions: $C_x(j) = lsb(jx) = 1$ *for even* x*'s, and* $C_x(j) = lsb(jx) = lsb(j)$ *for odd* x*'s. Moreover, no alternative list decoding algorithm exists, since each codeword is at distance 0 from half the codewords.*

5.5 Putting It Together

THEOREM 3. *Let* $\mathcal{P}$ *be a collection of segment predicates. Then,* $\mathcal{P}$ *is hard-core for* $RSA, Rabin, EXP, ECL$*, under the assumption that these are OWFs.*

6. SIMULTANEOUS SECURITY

DEFINITION 18 (HARD-CORE FUNCTION). *Let* $\mathcal{F} = \{\mathsf{f}_i : \mathcal{D}_i \to R_i\}_{i \in I}$ *be collections of OWFs. Let* $\mathcal{H} = \{\mathsf{h}_i : \mathcal{D}_i \to \{0,1\}^{l(i)}\}_{i \in I}$ *be collections of functions s.t. for each* $i \in I \cap \{0,1\}^k$, $l(i)$ *is polynomial in* k.

We say that a PPT algorithm D *is a* distinguisher *for* $\mathcal{H}$ *w.r. to* $\mathcal{F}$*, if* $\exists$ *a non-negligible function* ρ *s.t.*

$$|\Pr[D(\mathsf{f}_i(x), \mathsf{h}_i(x)) = 1] - \Pr[D(\mathsf{f}_i(x), \mathsf{h}(r)) = 1]| \geq \rho(k)$$

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

We say that $\mathcal{H}$ *is* hard-core *for* $\mathcal{F}$*, if there exists no distinguisher for* $\mathcal{H}$ *w.r. to* $\mathcal{F}$.

We now extend the definition of segment predicates to segment functions.

DEFINITION 19 (SEGMENT FUNCTION). *Let* $\mathcal{H} = \{\mathsf{h}_N : Z_N \to \{0,1\}^{l(N)}\}_{N \in I}$ *be a collection of functions. For each* $s \in \{0,1\}^{l(N)}$*, define a predicate* $P_N^{\mathcal{H},s} : Z_N \to \{0,1\}$, $P_N^{\mathcal{H},s}(x) = 1$ *if* $\mathsf{h}_N(x) = s$ *and 0 otherwise. We say that* $\mathcal{H}$ *is a collection of* segment functions*, if* $\mathcal{P} = \left\{P_N^{\mathcal{H},s}\right\}_{N \in I, s \in \{0,1\}^{l(N)}}$ *is a collection of segment predicates.*

It turns out that for proving simultaneous security it is useful to consider unbalanced segment predicates as in the following examples.

EXAMPLE 2 (MOST SIGNIFICANT BITS). *Let* $Pref_N(x)$ *be the* $l(N)$ *most significant bits in a binary representation of* x. $\forall s \in \{0,1\}^{l(N)}$*, define the unbalanced predicate* $P_N^s(x) = 1$ *if* $Pref_N(x) = s$*, and 0 otherwise. When* $l(N) \leq O(\log\log N)$, P_N^s *is is a segment predicate (as it is non-negligibly far from constant), thus,* $\mathcal{H} = \{Pref_N : Z_N \to \{0,1\}^{l(N)}\}_N$ *is a collection of segment functions.*

EXAMPLE 3 (DISSECTION BITS). *Let* $a \in Z_N^*$*, and let* $Dissect_{a,N}(x) = Pref_N(x/a)$*. Then when* $l(N) \leq O(\log\log N)$, $\mathcal{H} = \left\{Dissect_{a,N} : Z_N \to \{0,1\}^{l(N)}\right\}_N$ *is a collection of segment functions.*

THEOREM 4. *Let* $\mathcal{H} = \left\{\mathsf{h}_N : Z_N \to \{0,1\}^{l(N)}\right\}_N$ *be a collection of segment functions. Then* $\mathcal{H}$ *is hard-core for RSA, Rabin, EXP and ECL, under the assumption that these are OWFs.*

7. LEARNING HEAVY FOURIER COEFFICIENTS OF FUNCTIONS OVER ABELIAN GROUPS

In this section, we consider the problem of finding all the heavy Fourier coefficient of a given function g.

THEOREM 5 (LEARNING). *Let G be a finite abelian group with known set of generators of known orders. There is a learning algorithm that, given query access to $g: G \to \mathbb{C}$, $0 < \tau$ and $0 < \delta < 1$, outputs a list L, of $O(\|g\|_2^2/\tau)$ characters (each can be encoded by $\log|G|$ bits), that contains $\mathsf{Heavy}_\tau(g)$, w.p. at least $1-\delta$; and the running time of the algorithm is polynomial in $\log|G|, \|g\|_\infty, 1/\tau$ and $\ln(1/\delta)$.*

For the sake of this extended abstract, we will describe the restricted case of g being a Boolean function defined over Z_N. A description of the general case is deferred to the full version of this paper.

7.1 Related Learning Work

Our algorithm extends a previous algorithm by Kushilevitz and Mansour [14], which learns heavy Fourier coefficients of functions $g: \{0,1\}^k \to \{\pm 1\}$. The main difficulty in extending the algorithm of [14] is as follows. Consider the case of g over $\{0,...,N-1\}$, and identify each character with an element $\alpha_1...\alpha_k \in \{0,...,N-1\}^k$. A central step in the algorithm of [14] is fixing some prefix $\alpha^0 = \alpha_1...\alpha_{i-1}$ and checking for each $\alpha_i \in \{0,1\}$ whether the prefix $\alpha^0\alpha_i$ can be extended to a heavy character. When $\alpha_i \in \{0,...,N-1\}$, it is impossible to consider every possible value, since N is thought of as exponentially large. Instead, we devise an efficient search procedure for finding the few relevant α_i's.

We note that in [16] Mansour extended techniques from [14] to an algorithm that learns the heavy coefficients of a polynomial P, when given black-box query access to P. This latter algorithm can be interpreted as an algorithm that finds the heavy Fourier coefficients of a function $g: Z_N^k \to \mathbb{C}$, where N is restricted to be a power of 2.

7.2 The Boolean Function over Z_N Case

Let us state the result for Boolean functions over Z_N.

THEOREM 6. *There is an algorithm that, given query access to $g: Z_N \to \{\pm 1\}$, $0 < \tau$ and $0 < \delta < 1$, outputs a list L, of $O(1/\tau)$ characters (each can be encoded in $\log N$ bits), that contain $\mathsf{Heavy}_\tau(g)$, w.p. at least $1-\delta$; and the running time[5] of the algorithm is $\widetilde{O}(\log N \cdot \ln^2(1/\delta)/\tau^{5.5})$.*

7.2.1 Overview of the algorithm

The algorithm begins with an initial collection of intervals $\mathcal{C}_0$, obtained by a partition of Z_N into intervals of size $\frac{N}{l_0}$. Each interval in $\mathcal{C}_0$ is viewed as a candidate for containing some α such that χ_α is a heavy character of g.

The algorithm consists of $O(\log \frac{N}{l_0})$ steps of refining intervals' collections. In refining step $i \geq 1$ each interval in collection $\mathcal{C}_{i-1}$ is halved into two sub-interval, now of size $\frac{N}{l_0 \cdot 2^i}$. Each sub-interval is either inserted into a new collection $\mathcal{C}_i$ or discarded depending on the outcome of a procedure which distinguishes (with high probability) between intervals which contain some α for which χ_α is heavy and those which are "far from" containing any such α (in the sense that the sum of weights of all characters in a slightly larger interval is less than $c \cdot \tau$, for some constant $c < 1$). Indeed, the heart of the algorithm is to devise such an efficient distinguishing procedure (see refinement section 7.2.3 for details).

Once all refining steps are completed, the algorithm holds a final collection of singleton intervals including amongst them all heavy characters, and possibly some other characters as well. In a post-processing step, one may further shrink down this list of characters in the collection to coincide (with high probability) with a list of length $O(\frac{1}{\tau})$ containing all heavy characters of g.

7.2.2 The Algorithm

The notion of a *good l-collection* of intervals is central to our algorithm.

DEFINITION 20 (GOOD l-COLLECTION). *Let $J_j^l = [(j-1)\lceil (N/l)\rceil, j\lceil (N/l)\rceil - 1]$ denote an interval of size $\lceil (\frac{N}{l})\rceil$. An l-collection of intervals $\mathcal{C}$ is a subset of the intervals $\left\{J_j^l\right\}_{1\leq j\leq l}$. We say that an l-collection $\mathcal{C}$ is* good *if :*

1. *$|\mathcal{C}| \leq O(1/\tau^{1.5})$ ($\mathcal{C}$ is small), and*
2. *$\forall \chi_\alpha \in \mathsf{Heavy}_\tau(g)$, $\exists J \in \mathcal{C}$ such that $\alpha \in J$ (all heavy characters are included in the intervals of $\mathcal{C}$)*

Algorithm LearnHeavyCoefficients
Input: query access to $g: Z_N \to \{\pm 1\}$, $\tau > 0$ and $0 < \delta < 1$
Output: a list of characters L, such that L is of length at most $O(1/\tau)$, and it contains $\mathsf{Heavy}_\tau(g)$ w.p. at least $1-\delta$
Algorithm:

1. *Initialization:* Let $j = 0$, $l_0 = O(1/\sqrt{\tau})$; and $\mathcal{C}_0 = \left\{J_i^{l_0}\right\}_{i=1,...,l_0}$.
2. *Refinement:* repeat $\log(N/l_0)$ times: $\mathcal{C}_{j+1} = Refine(\mathcal{C}_j)$; $j \leftarrow j+1$; $l_j = 2l_{j-1}$.
3. Return $\mathsf{Shrink}(\mathcal{C}_j)$

Remarks:

1. Throughout the algorithm, for $1 \leq j \leq O(\log \frac{N}{l_0})$, we keep the invariant that C_j is a good l_j-collection. The search for heavy characters is initiated by setting $l_0 = O(\frac{1}{\sqrt{\tau}})$, and initializing the l_0-collection $C_0 = \left\{J_i^{l_0}\right\}_{i=1,...,l_0}$. This is a good collection: it is small since l_0 is small, and it contains all heavy characters since it covers Z_N. Next, at each refining step j, in time $\widetilde{O}(\frac{\ln^2(1/\delta)}{\tau^{5.5}})$ we refine the current good l_j-collection C_j into a good $l_{j+1} = 2l_j$-collection C_{j+1}, w.p. at least $1 - O(\frac{\delta}{\log N})$ (for details and proof see section 7.2.3). Finally, after $O(\log(\frac{N}{l_0}))$ steps, the final collection consists of $O(\frac{1}{\tau^{1.5}})$ intervals of length 1 each. This final collection is good w.p. at least $1 - O(\delta)$ by union bound over the refining steps.
2. The final collection, being good, thus constitutes a list of length at most $O(\frac{1}{\tau^{1.5}})$, which contains all heavy characters, w.p. at least $1 - O(\delta)$. To shrink this list, we estimate the weight of each character in the list, such that w.h.p. the distance of our estimation from the real value is no more than $\tau/4$ (this can be done efficiently, by Chernoff bound).

[5] We use $\widetilde{O}()$ notation to indicate that we omit terms of complexity which is polynomial in $\log(1/\tau), \log\log N$ or $\ln\ln(1/\delta)$.

We maintain only the characters estimated to be of weight at least $3\tau/4$. The shrunk list contains, w.h.p., all heavy characters, and no character of weight less than $\tau/2$. Thus, the final output is a list of length $O(\frac{1}{\tau})$, containing all heavy characters with probability at least $1-\delta$.

3. As our goal was to prove a polynomial upper bound, we did not bother optimizing the complexity of the learning algorithm. However, by a more careful analysis and choice of parameters, its complexity can be improved. This has significance for a tighter security proof for the hard-core predicates and more importantly for devising efficient algorithms for discovering heavy Fourier coefficients in applications involving FFT computations.

7.2.3 *Refinement Procedure*

In a refinement step, we efficiently transform a good l-collection $\mathcal{C}$ into a good $2l$-collection $\mathcal{C}'$ as follows. Let the intervals that are candidates for $\mathcal{C}'$ be $Candidate = \{J_i^{2l}$ that intersects an interval in $\mathcal{C}\}_{1\le i\le 2l}$. It is sufficient to consider whether to keep or discard these intervals, as each heavy character of g is contained in some interval of $\mathcal{C}$. Namely, we consider at most $3|\mathcal{C}| \le O(1/\tau^{1.5})$ intervals (by the invariant of $\mathcal{C}$ being small).

Ideally, we would like now to discard all candidate intervals that do not contain a heavy character. However, we do not know how to efficiently decide if an interval contains a heavy character or not. Instead we discard all interval that are "far from" containing a heavy character:

DEFINITION 21 (FAR FROM HEAVY). *For each interval J, denote $weight(J) = \sum_{\alpha\in J}|\widehat{g}(\alpha)|^2$. For each J_i^{2l} let*

$$Ext(J_i^{2l}) = \bigcup_{j=-\Delta}^{\Delta} J_{i+j}^{2l},\ \text{for } \Delta = \lceil(2\sqrt{24/\tau})\rceil$$

We say that J_i^{2l} is far from heavy *if $weight(Ext(J_i^{2l})) < \tau/24$.*

We devise a *distinguishing* procedure to distinguish between (1) the case that an interval $J = J_i^{2l}$ contains an α such that χ_α is heavy, from the case (2) J is far from heavy. In the latter case, we discard the interval J, while otherwise, we insert J to $\mathcal{C}'$. Clearly, we lose nothing, as for every $\alpha \in J$, the weight of $\chi_\alpha \le weight(J) \le weight(Ext(J)) \le \tau$.

Refine Procedure
Input: a good l-collection $\mathcal{C}$
Output: a good $2l$-collection $\mathcal{C}'$
Algorithm:

1. *Initialization*: $Candidates = \{J_i^{2l}$ that intersect some $J \in \mathcal{C}\}_{i=1,\ldots,2l}$; $\mathcal{C}'$ is the empty collection.
2. $\forall J_i^{2l} \in Candidates$, if Distinguishing$(J_i^{2l})$ = Yes, $\mathcal{C}' \leftarrow \mathcal{C}' \cup \{J_i^{2l}\}$
3. Return $\mathcal{C}'$.

LEMMA 8 (REFINE). *Given a good l-collection $\mathcal{C}$, w.p. at least $1 - O(\delta/\log N)$, the refinement procedure returns a good $2l$-collection $\mathcal{C}'$; and its running time is $O(T/\tau^{1.5})$, for $T = \widetilde{O}(\ln^2(1/\delta)/\tau^4)$ the running time of the distinguishing procedure.*

7.2.4 *Distinguishing Procedure*

The distinguishing procedure is the heart of the algorithm.
Distinguishing *Procedure*
Input: an interval J_i^{2l}
Output: Yes/No
Algorithm:

1. Let $\varepsilon = \delta\tau^{1.5}/\log N$, $m_2 = \Theta(\ln(1/\varepsilon)/\tau^2)$, $m_1 = \Theta(\ln(m_2/\varepsilon)/\tau^2)$. Randomly choose $x_1, \ldots, x_{m_2}$ samples in Z_N
2. For each x_r, $r \in 1, \ldots, m_2$
   - Randomly choose $y_1, \ldots, y_{m_1}$ samples in $\{0, \ldots, l-1\}$
   - Let $\mathsf{g}^i(x_r) = \chi_{shift}(x_r) \cdot \frac{1}{m_1}\sum_{t=1}^{m_1} \mathsf{g}(x_r - y_t)$, for $shift = -(i-1)\lceil(N/2l)\rceil$
3. Let $\mathrm{est} = \frac{1}{m_2}\sum_{r=1}^{m_2} \mathsf{g}^i(x_r)^2$
4. If est $\ge \tau/8$ return Yes, otherwise, return No

LEMMA 9 (DISTINGUISHER). *Given query access to g: $Z_N \to \{\pm 1\}$, $\tau, \delta > 0$, and an interval J_i^{2l}, the* Distinguishing *procedure above runs in time $T = \widetilde{O}(\ln^2(1/\delta)/\tau^4)$, and w.p. at least $1 - O(\delta\tau^{1.5}/\log N)$, returns*

1. Yes, *when J_i^{2l} contains a heavy character*
2. No, *when J_i^{2l} is far from heavy*

Proof. We distinguish between the above cases by evaluating $weight(J_i^{2l})$. Ideally, we would like to compute $weight(J_i^{2l})$, by computing the norm of the restriction $\|\mathsf{g}_{|\{\chi_\alpha:\ \alpha\in J_i^{2l}\}}\|_2^2 = weight(J_i^{2l})$. But, how do we access $\mathsf{g}_{|\{\chi_\alpha:\ \alpha\in J_i^{2l}\}}$, and how do we compute its norm?

Decaying function: Relaxing the requirement of accessing $\mathsf{g}_{|\{\chi_\alpha:\ \alpha\in J_i^{2l}\}}$, we would like to access a J_i^{2l}-decaying function h^i defined as follows. Let $\mathsf{d}(\alpha, J_i^{2l})$ denote the distance of $\alpha \in Z_N$ from the leftmost point in J_i^{2l}. We say that a function h^i is J_i^{2l}*-decaying*, if $\left|\widehat{\mathsf{h}^i}(\alpha)\right| = c(\alpha)\,|\widehat{\mathsf{g}}(\alpha)|$ for $0 \le c(\alpha) \le 1$ s.t. for $\alpha \in J_i^{2l}$, $c(\alpha)$ is fairly high, *i.e.*, $\Omega(1) \le c(\alpha)$; and for $\alpha \notin J_i^{2l}$, $c(\alpha)$ decreases rapidly, *i.e.*, $c(\alpha) \le O(|J_i^{2l}|/\mathsf{d}(\alpha, J_i^{2l}))$.

Being able to compute $\sum_{\alpha\in Z_N}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2$ provides a good handle on $weight(J_i^{2l})$ since,

$$\Omega(weight(J_i^{2l})) \le \sum_{\alpha\in Z_N}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2 \le O(weight(Ext(J_i^{2l})))+O(\tau)$$

follows from

- $\sum_{\alpha\in J_i^{2l}}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2 \ge \Omega(weight(J_i^{2l}))$.
- $\sum_{\alpha\in Ext(J_i^{2l})}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2 \le O(weight(Ext(J_i^{2l})))$
- $\sum_{\alpha\notin Ext(J_i^{2l})}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2 \le O(\tau)\cdot\sum_{\alpha\in Z_N}|\widehat{\mathsf{g}}(\alpha)|^2 = O(\tau)$ (as for Boolean g's, $\sum_{\alpha\in Z_N}|\widehat{\mathsf{g}}(\alpha)|^2 = 1$).

In the following, we exhibit a J_i^{2l}-decaying function h^i; show how to estimate $\sum_{\alpha\in Z_N}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2$; and in turn, use it to evaluate $weight(J_i^{2l})$.

Let us define the J_i^{2l}-decaying function h^i. First, we address J_1^{2l}. The function

$$\mathsf{h}^1(x) = \mathop{\mathbb{E}}_{y=0,\ldots,l-1}[\mathsf{g}(x-y)]$$

is J_1^{2l}-decaying, since $\left|\widehat{\mathsf{h}^1}(\alpha)\right| = c(\alpha)\left|\widehat{\mathsf{g}}(\alpha)\right|$, for $c(\alpha) = \left|\mathbb{E}_{y=0,\ldots,l-1}[\chi_\alpha(y)]\right|$, where, $0 \le c(\alpha) \le 1$, and by proposition 1, $c(\alpha) \ge 1/6$ for $\alpha \in J_1^{2l}$, and $c(\alpha) \le 2\left|J_1^{2l}\right|/\mathsf{abs}(\alpha)$ otherwise.

Second, we address J_i^{2l}. Let $shift = -(i-1)\lceil (N/2l)\rceil$ denote the offset for shifting J_i^{2l} to J_1^{2l}, then

$$\mathsf{h}^i(x) = \chi_{shift}(x) \cdot \mathop{\mathbb{E}}_{y=0,\ldots,l-1}[\mathsf{g}(x-y)]$$

is J_i^{2l}-decaying, since $\left|\widehat{\mathsf{h}^i}(\alpha)\right| = \left|\widehat{\mathsf{h}^1}(shift+\alpha)\right|$ where h^1 is J_1^{2l}-decaying, and $\mathsf{d}(\alpha, J_i^{2l}) = \mathsf{abs}(\alpha + shift)$. *I.e.*, $\left|\widehat{\mathsf{h}^i}(\alpha)\right| = c(\alpha)\left|\widehat{\mathsf{g}}(\alpha)\right|$ for $0 \le c(\alpha) \le 1$ s.t. $c(\alpha) \ge 1/6$ for $\alpha \in J_i^{2l}$, and $c(\alpha) \le 2\left|J_i^{2l}\right|/\mathsf{d}(\alpha, J_i^{2l})$ otherwise.

We proceed by showing how to estimate $\sum_{\alpha\in Z_N}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2$. By Parseval identity, $\sum_{\alpha\in Z_N}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2 = \mathbb{E}_{x\in Z_N}[\left|\mathsf{h}^i(x)\right|^2]$. Thus, we need to estimate $\mathbb{E}_x[\left|\mathsf{h}^i(x)\right|^2]$.

Denote the confidence of the distinguishing procedure by $\varepsilon = \delta\tau^{1.5}/\log N$. Let $\eta = \tau/48$, $m_2 = \Theta(\ln(1/\varepsilon)/\eta^2)$ and $m_1 = \Theta(\ln(m_2/\varepsilon)/\eta^2)$. Let $x_1, \ldots, x_{m_2} \in_R Z_N$ be random samples, then by Chernoff bound, w.p. at least $1 - O(\varepsilon)$, $\left|\frac{1}{m_2}\sum_{r=1}^{m_2}\left|\mathsf{h}^i(x_r)\right|^2 - \sum_\alpha\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2\right| \le \eta$. For estimating a single value $\mathsf{h}^i(x)$, let $y_1, \ldots, y_{m_1} \in_R \{0, \ldots, l-1\}$ be random samples, then by Chernoff bound,

$$\mathsf{g}^i(x) = \chi_{shift}(x) \cdot \frac{1}{m_1}\sum_{t=1}^{m_1}\mathsf{g}(x-y_t)$$

satisfies $\left|\mathsf{g}^i(x) - \mathsf{h}^i(x)\right| < \eta$, w.p. at-least $1 - O(\varepsilon/m_2)$. Furthermore, by union bound,

$$\mathsf{est} = \sum_{r=1}^{m_2}\left|\mathsf{g}^i(x_r)\right|^2$$

satisfies that $\left|\mathsf{est} - \frac{1}{m_2}\sum_{r=1}^{m_2}\left|\mathsf{h}^i(x_r)\right|^2\right| < \eta$, w.p. at least $1 - O(\varepsilon)$. Combined together, w.p. at least $1 - O(\varepsilon)$, $\left|\mathsf{est} - \sum_{\alpha\in Z_N}\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2\right| \le 2\eta$.

Finally, we utilize the estimation of $\sum_\alpha\left|\widehat{\mathsf{h}^i}(\alpha)\right|^2$ to evaluate $weight(J_i^{2l})$. Recall that for $\alpha \in J_i^{2l}$, $c(\alpha) > 1/6$, and $2\eta = \tau/24$. By the definition of $Ext(J_i^{2l})$, for $\alpha \notin Ext(J_i^{2l})$, $c(\alpha) \le \tau/24$. Therefore, on the one hand, $weight(J_i^{2l}) \ge \tau$ implies

$$\mathsf{est} > weight(J_i^{2l})/6 - 2\eta \ge \tau/8$$

and on the other hand, $weight(Ext(J_i^{2l})) < \tau/24$ implies

$$\mathsf{est} < weight(Ext(J_i^{2l})) + \tau/24 + 2\eta \le \tau/8$$

Hence, by answering Yes iff $\mathsf{est} \ge \tau/8$, we distinguish the two cases of the lemma, w.p. at least $1 - O(\varepsilon) = 1 - O(\delta\tau^{1.5}/\log N)$; and the running time of the distinguishing procedure is $O(m_1 \cdot m_2) = \widetilde{O}(\ln^2(1/\delta)/\tau^4)$. □

8. ACKNOWLEDGEMENTS

We are grateful to Gady Kozma for help with the proof of the concentration and agreement lemma. We are very thankful to Alon Rosen, Oded Goldreich, Madhu Sudan, Zeev Rudnick, Par Kurlberg, Yishay Mansour and Guy Kindler for very useful discussions and comments on this work.

9. REFERENCES

[1] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *SIAM J. Computing*, 17:194–209, 1988.

[2] M. Ben-Or, B. Chor, and A. Shamir. On the cryptographic security of single RSA bits. In *In Proc. 15th ACM Symposium on Theory of Computing*, pages 421–430, 1983.

[3] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.

[4] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[5] R. Fischlin and P. Schnorr. Stronger security proofs for RSA and Rabin bits. *J. Cryptology*, 13:221–244, 2000.

[6] M. Goldmann and A. Russell. Spectral bounds on general hard core predicates. In *In Proc. of STACS*, pages 614–625, 2000.

[7] O. Goldreich and L. Levin. A hard-core predicate for all one-way functions. In *STOC*, 1989.

[8] Oded Goldreich. *Foundations of Cryptography, Basic Tools.* Cambridge university press, 2001.

[9] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, 28(2):270–299, 1984.

[10] S. Goldwasser, S. Micali, and P. Tong. Why and how to establish a private code on a public network. In *In Proc. 23rd IEEE Symp. on Foundations of Comp. Science*, pages 134–144, 1982.

[11] J. Hastad and M. Naslund. The security of individual RSA bits. In *IEEE Symposium on Foundations of Computer Science*, pages 510–521, 1998.

[12] R. Impagliazzo. private communication.

[13] B. S. Kaliski. A pseudo-random bit generator based on elliptic logarithms. In *In Advances in Cryptology — CRYPTO '86, A. M. Odlyzko, Ed., vol. 263 of Lecture Notes in Computer Science, Springer-Verlag*, volume 263, pages 84–103, 1986.

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

[15] D.L. Long and A. Wigderson. The Discrete Log problem hides $O(\log N)$ bits. *SIAM J. Comp.*, 17:363–372, 1988.

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

[17] M. Naslund. All bits in ax + b modp are hard (extended abstract). In *In Proc. of CRYPTO '96*, pages 114–128, 1996.

[18] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[19] M. Sudan. List decoding: Algorithms and applications. *SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory)*, 31, 2000.

[20] U.V. Vazirani and V.V. Vazirani. Efficient and secure pseudo-random number generation. In *Proc. 25th IEEE Symp. on Foundations of Computer Science*, pages 458–463, 1984.

[21] H. Wee. private communication.

[22] A.C. Yao. Theory and application of trapdoor functions. In *In 23rd IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.