

Table of Contents

Lecture 1: Locally Testable Codes — Introduction

1.1	Error correcting codes	1-1
1.2	Testers	1-1

Lecture 2: Hadamard codes are locally testable

2.1	Stating the BLR Theorem	2-1
2.2	Proving the BLR Theorem	2-2

Lecture 3: Fourier Analysis of Boolean Functions

3.1	Viewing words as points in \mathbb{R}^{2^k}	3-1
3.2	The Fourier basis and the set of characters	3-1

Lecture 4: Better soundness for Hadamard via Fourier Analysis

4.1	Fourier analysis of the BLR-tester	4-1
4.2	Query-efficient boosting of soundness via Graph-tests	4-1
4.3	Proof of Theorem 4.2	4-2

Lecture 5: Testing Reed-Solomon and bivariate Reed-Muller codes

5.1	Reed-Solomon codes	5-1
5.2	Bivariate low-degree testing	5-2

Lecture 6: Bivariate low degree testing

6.1	Trading degree for errors	6-1
6.2	Bivariate Division	6-2

Lecture 7: Bivariate division

7.1	Completing the proof of Theorem 6.1	7-1
7.2	Proving Lemma 6.2	7-1

Lecture 8: Bivariate factoring

8.1	The Resultant	8-1
8.2	The Resultant polynomial	8-1
8.3	Finding a nontrivial common factor	8-2
8.4	Reducing the number of resultant roots	8-2
8.5	Suggested research projects regarding LTCs	8-3

Lecture 9: Lattices — Introduction

Lecture 10: The LLL algorithm

10.1	Reduced basis	10-1
10.2	The LLL Algorithm	10-2
10.3	Bounding the runtime of the algorithm	10-2

Lecture 11: A lattice based attack on the RSA cryptosystem

11.1	Low Exponent Attack	11-1
11.2	Reducing Theorem 11.2 to Theorem 11.3	11-2
11.3	Proof of Theorem 11.3	11-3

Lecture 12: Computational complexity of basic lattice problems

12.1	Decisional CVP is NP -complete	12-1
12.2	The complexity of approximate SVP	12-2
12.3	Complexity of SVP — State of the art	12-2

References

1.1 Error correcting codes

Error correcting codes were devised for communicating information over noisy channels. Since their introduction in the 1940's in the pioneering work of Golay, Hamming and Shannon they have found numerous applications in Computer Science, Electrical Engineering and Combinatorics. For an introduction to error correcting codes see e.g. [MacWilliams and Sloane \[1977\]](#); [Berlekamp \[1984\]](#).

Definition 1.1 (Error-Correcting Code). Let F be a finite field and $k, n, d \in \mathbb{N}^+$. An *error correcting code* or simply a *code* C over F is an injective mapping $E_C : F^k \rightarrow F^n$ called the *encoding* of C . The *code* or set of *codewords* is

$$C = \{E_C(x) \mid x \in F^k\} \subseteq F^n.$$

The *minimal distance*, or simply *distance* of C is

$$d = n \cdot \min \{\Delta(w, w') \mid w' \in C, w' \neq w\},$$

where Δ denotes the *relative Hamming distance* between $w = (w_1, \dots, w_n)$ and $w' = (w'_1, \dots, w'_n)$

$$\Delta(w, w') = \Pr_{i \in [n]} [w_i \neq w'_i]$$

If E_C is a linear mapping then C is said to be a $[n, k, d]_F$ -LECC (*linear error correcting code*).

Basic parameters of a code n is the *block length* of the code; k is the *dimension* of C ; k/n is the code's *rate* and d/n is its *relative* or *normalized distance*.

Problem 1.1. The *support* of $w \in F^n$ is $\text{supp}(w) = \{i \in [n] \mid w_i \neq 0\}$. The *weight* of w is the size of its support. Prove: The distance of a linear code is equal to the minimal *weight* of a nonzero codeword.

1.2 Testers

We wish to decide whether an input $w \in F^n$ is a codeword of C , while minimizing the computational resources needed for the task. An “ideal” *tester* would run in zero time and always make correct decisions. Sadly, an ideal tester does not exist. Therefore, we try to construct testers that come as close as possible to the ideal. In particular, we try to minimize resources such as running time and number of queries made to the input. Surprisingly, for a number of interesting codes we'll present *sublinear* testers, i.e., the running time and

number of queries will be smaller than the code's blocklength. Notice that such super-efficient testing forbids even a single pass over the input.

We also try to maximize the probability that the tester's decision is correct. To measure the "success rate" of a tester we define its *completeness* to be the minimal probability that a good word is accepted by it and the *soundness* as the minimal probability that a bad word is rejected. An ideal tester would have *perfect* completeness and soundness, i.e., both parameters would be equal to 1. All testers we'll consider will have nonperfect soundness meaning some noncodewords will be erroneously labeled as codewords. Most testers we'll see will have perfect completeness, i.e., good codewords will always be labeled as such.

Intuitively, the farther a received word w is from the code, the easier it should be for the tester to reject it. Thus, we will define the soundness of a tester to be a function of the Hamming distance of w from C :

$$\Delta(w, C) = \min \{ \Delta(w, w') \mid w' \in C \}.$$

The notion of a randomized tester restricted to use only sublinear computational resources is relatively new. Variants of the following definition are implicit in [Blum et al. \[1993\]](#); [Babai et al. \[1991\]](#) and appeared explicitly in [Friedl and Sudan \[1995\]](#); [Rubinfeld and Sudan \[1996\]](#); [Arora \[1994\]](#); [Spielman \[1995\]](#).

Definition 1.2 (Tester). Let C be a code with blocklength n over field F . A (t, r, q) -restricted tester for C is a randomized Turing machine T with oracle access to an input $w \in F^n$ that

- Runs in time $\leq t$
- Tosses $\leq r$ random coins
- Queries $\leq q$ entries of w where each entry is an element of F
- Outputs either **accept** or **reject**

Let $T^w[R] \in \{\text{accept}, \text{reject}\}$ denote the output of the tester T on random coins $R \in \{0, 1\}^r$ and input w . We say T has *completeness* $c \in [0, 1]$ and *soundness* $s : [0, 1] \rightarrow [0, 1]$ if

- **Completeness:** $w \in C$ implies $\Pr_{R \in \{0, 1\}^r} [T^w[R] = \text{accept}] \geq c$.
- **Soundness:** $w \notin C$ implies $\Pr_{R \in \{0, 1\}^r} [T^w[R] = \text{accept}] \leq 1 - s(\Delta(w, C))$.

A tester is said to be *nonadaptive* if the set of queries it makes and the computation performed on the answers it receives depends only on the randomness R . A nonadaptive tester uses its randomness R to select a set of indices $I = I_R = \{i_1, \dots, i_q\} \subseteq [n]$ and a decision predicate $D = D_R : F^I \rightarrow \{\text{accept}, \text{reject}\}$ and outputs $T^w[R] = D(w_{i_1}, \dots, w_{i_q})$.

Problem 1.2. Suppose T is a nonadaptive tester for a code with blocklength n and distance d . Suppose furthermore T has *perfect* completeness $c = 1$ and makes q queries in each run. Let $\delta = \Delta(w, C)$. Bound from above the soundness of T , i.e., prove an upper bound on $s(\delta)$ for $\delta < d/2n$.

Problem 1.3. Let C be a $[n, k, d]_F$ -LECC. Suggest a nonadaptive tester with perfect completeness and small query complexity for C based on the dual code

$$C^\perp = \{u \in F^n \mid \langle u, v \rangle = 0 \text{ for all } v \in C\},$$

where $\langle (v_1, \dots, v_n), (u_1, \dots, u_n) \rangle = \sum_{i=1}^n v_i \cdot u_i$.

In this session we will prove that the family of Hadamard codes is locally testable. This result, due to Blum, Luby and Rubinfeld, was the first example of a locally testable code and has had a great influence on our subject. We'll see two proofs of the local testability of Hadamard codes. The first one will be somewhat weaker in terms of its soundness function, however it can be generalized to other settings. This proof method is due to Coppersmith. It appeared in [Blum et al. \[1993\]](#) and is a simplification of the original proof found by the authors (this original proof was not published). The second proof method, the topic of our next session, was introduced in [Bellare et al. \[1996\]](#). It is based on Fourier analysis and gives better bounds on the soundness function, however, its generalizations are more limited.

2.1 Stating the BLR Theorem

Let F_q denote the finite field with q elements. A function $f : F_q^k \rightarrow F_q$ is said to be F_q -linear, or simply linear, if

$$f(a \cdot x + b \cdot y) = a \cdot f(x) + b \cdot f(y), \quad \text{for all } x, y \in F_q^k, a, b \in F_q.$$

Problem 2.1. Prove: A function $f : F_2^k \rightarrow F_2$ is linear iff there exists $x \in F_2^k$ such that $f(m) = \langle m, x \rangle$ for all $m \in F_2^k$.

Definition 2.1 (Hadamard codes). For $k \in \mathbb{N}^+$, the *Hadamard code* of dimension k is the $[2^k, k, 2^{k-1}]_{F_2}$ -LECC in which a message $m \in F_2^k$ is encoded by its evaluation by all F_2 -linear functions. Formally, the encoding of m is of length 2^k and has one entry for each $x \in F_2^k$. The value of the x^{th} entry, denoted w_x , is $\langle m, x \rangle$. We use H_k to denote the k -dimensional *Hadamard code*, i.e. the set of its codewords.

Problem 2.2. Prove: the minimal distance of the k -dimensional Hadamard code is 2^{k-1} .

The following tester for the Hadamard code was suggested by [Blum et al. \[1993\]](#).

Definition 2.2 (BLR-Tester). For $k \in \mathbb{N}^+$ let T_{H_k} be the tester operating as follows when given oracle access to a supposed Hadamard codeword $w \in F_2^{2^k}$:

- Use $2k$ random bits to select $x, y \in F_2^k$.
- Let $z = x + y$ (addition of vectors in F_2^k).
- Output **accept** iff $w_x + w_y + w_z = 0$ and otherwise output **reject**.

Let $T_{H_k}^w[x, y]$ denote the output of the tester when x, y are the random elements selected in the first step.

Problem 2.3. Formulate and prove: The BLR-tester has perfect completeness.

We shall prove the following bound on the soundness function, shown originally by [Blum et al. \[1993\]](#).

Theorem 2.3. For all $k \in \mathbb{N}^+$ letting T denote the BLR tester T_{H_k} ,

$$\text{If } \Pr_{x,y \in F_2^k} [T^w[x, y] = \text{reject}] = \varepsilon < 2/9, \quad \text{Then } \Delta(w, H_k) \leq 2\varepsilon.$$

Problem 2.4. Use the previous theorem to bound from below the soundness function of the BLR-Tester.

2.2 Proving the BLR Theorem

For the rest of this session we assume $\Pr[T^w = \text{reject}] = \varepsilon < 2/9$. Under this assumption we shall (i) present a word w' that is 2ε -close to w and (ii) prove $w' \in H_k$. The first part is relatively easy to prove and the second part is somewhat harder.

Definition 2.4 (Majority decoding). Given $w \in F_2^{2k}$ let $w' \in F_2^{2k}$ be defined by setting the x^{th} entry of w' to be the most common value of $w_{x+y} + w_y$. Formally,

$$w'_x = \text{majority} \left\{ w_{x+y} + w_y \mid y \in F_2^k \right\}.$$

Problem 2.5. Prove part (i) above: $\Delta(w, w') \leq 2 \cdot \Pr_{x,y} [T^w[x, y] = \text{reject}]$. Hint: Let $B \subset F_2^k$ be the set of bad entries for which $w_x \neq w'_x$ and show an upper bound on $|B|$.

For the rest of this session we'll focus on part (ii) and prove $w' \in H_k$. Our first step is to show that every entry w'_x is selected by an overwhelming majority.

Problem 2.6. Prove: For all $x \in F_2^k$,

$$\Pr_y [w_{x+y} + w_y = w'_x] > 2/3.$$

Hints:

- Consider $y, z \in F_2^k$ selected randomly and independently.
- Show $\Pr_{y,z} [w_{x+y} + w_z = w_{x+z} + w_y] = \Pr_{y,z} [w_{x+y} + w_y = w_{x+z} + w_z] > 5/9$.
- Notice $\Pr_{y,z} [w_{x+y} + w_y = w_{x+z} + w_z]$ is the probability that two independent random "votes" for w'_x agree. The probability that two independent identically distributed random variables agree on their value is the *collision probability* of the distribution and is often useful for various probabilistic arguments, as we'll see next.
- Prove: If a $\{0, 1\}$ -valued random variable Y satisfying (w.l.o.g.) $\Pr[Y = 1] \geq 1/2$ has collision probability greater than $5/9$, then $\Pr[Y = 1] > 2/3$.

Problem 2.7. Using the previous problem prove that for any $x, y \in F_2^k$ we have $w'_x + w'_y = w'_{x+y}$. Conclude $w' \in H_k$.

Problem 2.8. Can [Theorem 2.3](#) be generalized? Does it hold when we replace F_2 by an arbitrary finite field F ? (Preliminary questions: What is the analogue of the Hadamard code over F ? What is the analogue of the BLR tester?) Can the generalization be pushed further than fields? (Surprising answer: yes!)

Last session we saw in [Problem 2.4](#) that the soundness of the BLR-tester for Hadamard codes satisfies $s(\delta) \geq \min\{\delta/2, 2/9\}$. This session we shall show better bounds for the BLR-tester. Our proof, due to [Bellare et al. \[1996\]](#), uses Fourier analysis to cast our problem as a geometric one. As a result we show $s(\delta) \geq \delta$ as stated in the following theorem from [Bellare et al. \[1996\]](#).

Theorem 3.1 (Improved BLR soundness). *For all $k \in \mathbb{N}^+$ letting T denote the BLR tester T_{H_k} ,*

$$\Pr_{x,y \in F_2^k} [T^w[x, y] = \text{accept}] \leq 1 - \Delta(w, H_k).$$

Proof Outline

- View the received word w as a point $f_w \in \mathbb{R}^{2^k}$, where \mathbb{R}^{2^k} is the 2^k -dimensional space over the reals.
- View the set of Hadamard codewords as an orthonormal basis for \mathbb{R}^{2^k} . This basis is known as the *Fourier basis* and its elements are called *characters*.
- Measure both $\Delta(w, H_k)$ and $\Pr_{x,y \in F_2^k} [T^w[x, y] = \text{accept}]$ in terms of the *Fourier representation* of f_w , i.e., its representation in the Fourier basis.

3.1 Viewing words as points in \mathbb{R}^{2^k}

Throughout this session we “translate” words $w \in F_2^{2^k}$ into functions f_w mapping the k -dimensional vector space F_2^k to $\{-1, 1\} \subset \mathbb{R}$,

$$f_w : F_2^k \rightarrow \{-1, 1\}, \quad f_w(x) = \begin{cases} 1 & w_x = 0 \\ -1 & w_x = 1 \end{cases}$$

Problem 3.1. Let \mathcal{F}_k denote the set of functions with domain F_2^k and range \mathbb{R} . Prove: \mathcal{F}_k is a vector space over \mathbb{R} . What is its dimension? Suggest a “canonical” basis for it.

3.2 The Fourier basis and the set of characters

We are going to prove that the set of “translations” of the Hadamard codewords form a basis for \mathcal{F}_k . Recall $|H_k| = 2^k$ so we have the right number of elements in our set. To prove this set is a basis we recall some basic linear algebra.

Inner products, bases, norms Define the *inner product* over the vector space \mathcal{F}_k with *normalization constant* $c > 0$ as

$$\langle f, g \rangle = c \cdot \sum_{x \in F_2^k} f(x) \cdot g(x), \quad f, g \in \mathcal{F}_k.$$

We say f and g are *orthogonal* if $\langle f, g \rangle = 0$. An inner product induces a *norm* function over the vector space \mathcal{F}_k ,

$$\|f\| = \sqrt{\langle f, f \rangle}.$$

We say f is *normal* if $\|f\| = 1$. A set of vectors $B = \{g_1, \dots, g_{2^k}\}$ is called *orthonormal* if all its elements are normal and every pair of distinct elements is orthogonal. Such a set forms an *orthonormal basis* for \mathcal{F}_k . If B is a basis, we can represent every element $f \in \mathcal{F}_k$ as a weighted sum of basis elements:

$$f = \sum_{i=1}^{2^k} \hat{f}_i \cdot g_i, \quad \hat{f}_i \in \mathbb{R}. \quad \text{I.e., } f(x) = \sum_{i=1}^{2^k} \hat{f}_i \cdot g_i(x) \text{ for all } x \in F_2^k.$$

Moreover, we can calculate the i^{th} coefficient in this representation by *projecting* f onto g_i , i.e., by taking the inner product:

$$\hat{f}_i = \langle f, g_i \rangle.$$

Finally, recall the norm is invariant under orthonormal basis transformations. Formally, if $\{g_1, \dots, g_{2^k}\}$, $\{h_1, \dots, h_{2^k}\}$ are two orthonormal bases and $\sum_i \hat{f}_i \cdot g_i$ and $\sum_i \tilde{f}_i \cdot h_i$ are two representations of (the same) f in these bases, then $\|\hat{f}\| = \|\tilde{f}\|$.

Problem 3.2. For $\alpha \in F_2^k$, let $\chi_\alpha : F_2^k \rightarrow \{-1, 1\}$ denote the “translation” of the Hadamard encoding of α . Such a function is called a *character*. Find an expression for calculating $\chi_\alpha(x)$. What is the function $\chi_{\vec{0}}$?

Remark. A *character* is a homomorphism from a group G to the complex roots of unity. In our case G is the additive group $\{F_2^k, +\}$. Because this group has order 2, i.e., $g + g = 0$ for all $g \in G$, it is not hard to prove that any homomorphism to the complex roots of unity must have its image either $\{-1, 1\}$ or $\{1\}$. Characters are very powerful tools used e.g. in Group Theory and Number Theory. (See [Serre \[1977\]](#) for an introduction to their use in Group Theory and [Lidl and Niederreiter \[1997\]](#) for some of their applications in Finite fields and Number Theory). Our proof exemplifies the use of characters in Theoretical Computer Science.

Problem 3.3. Prove the *bilinearity* of characters: $\chi_\alpha(x) \cdot \chi_\beta(x) = \chi_{\alpha+\beta}(x)$ for all $\alpha, \beta \in F_2^k$. Similarly, $\chi_\alpha(x) \cdot \chi_\alpha(y) = \chi_\alpha(x+y)$ for all $x, y \in F_2^k$.

Problem 3.4. (i) Prove: the set of characters is orthogonal. (ii) Find a normalization constant c for the inner product such that the set of characters forms an orthonormal basis for \mathcal{F}_k .

Notation The basis formed by the set of characters is called the *Fourier basis*. The representation $f = \sum_{\alpha \in F_2^k} \hat{f}_\alpha \cdot \chi_\alpha$ of a function $f \in \mathcal{F}_k$ is called the *Fourier representation* of f and $\hat{f}_\alpha = \langle f, \chi_\alpha \rangle$ is the α -*Fourier coefficient* of f .

Problem 3.5. Prove Parseval's identity:

$$\text{If } f : F_2^k \rightarrow \{-1, 1\}, \text{ Then } \sum_{\alpha \in F_2^k} \hat{f}_\alpha^2 = 1.$$

In this session we use the Fourier representation of Boolean functions introduced last session to get better bounds on the soundness of the BLR-Tester from [Definition 2.2](#). We'll start by proving [Theorem 3.1](#) using this approach. Then we'll use it to get better soundness via repeated, yet dependent invocations of the basic BLR-tester.

4.1 Fourier analysis of the BLR-tester

Once we cast our problem in the language of characters and Fourier-basis, the proof of [Theorem 3.1](#) reduces to a sequence of equation manipulations. Details follow. We continue to use the notation introduced last session.

Problem 4.1. Express $\Delta(w, H_k)$ in terms of the Fourier coefficients of f_w .

Problem 4.2. Express the success probability of the BLR-tester, i.e., $\Pr_{x,y}[w_x + w_y = w_{x+y}]$ in terms of the expectation of the three-term product $f_w(x) \cdot f_w(y) \cdot f_w(x+y)$.

Problem 4.3. Prove: For any boolean function $f : F_2^k \rightarrow \mathbb{R}$,

$$\mathbf{E}_{x,y}[f(x)f(y)f(x+y)] = \sum_{\alpha \in F_2^k} \hat{f}_\alpha^3.$$

- Express f in the Fourier basis.
- Use (i) linearity of expectation, (ii) orthogonality of characters, (iii) bilinearity of characters and (iv) independence of the variables x and y .

Problem 4.4. Prove [Theorem 3.1](#) using the previous problem and Parseval's identity.

4.2 Query-efficient boosting of soundness via Graph-tests

[Theorem 3.1](#) shows that if w is δ -far from H_k then at the price of 3 queries, the rejection probability of T^w is at least δ . A natural way to increase soundness, i.e., reduce the probability that w is accepted, is to repeat the BLR-test ℓ times. This incurs $q = 3\ell$ queries and reduces the acceptance probability of w from $1 - \delta$ to $(1 - \delta)^\ell \leq e^{-\delta\ell} = e^{-\delta\ell/3}$. Can we achieve similar error-reduction using fewer queries?

To answer this question, notice [Theorem 3.1](#) shows

$$\Pr_{x,y}[T^w[x,y] = \text{accept}] \leq \frac{1}{2} + \max \left\{ \hat{f}_\alpha \mid \alpha \in F_2^k \right\}$$

[Trevisan \[1998\]](#) suggested a query efficient *graph-tester* for reducing the acceptance probability exponentially in the number of queries q . [Samorodnitsky and Trevisan \[2000\]](#) showed that using this tester, the first summand above decreases from $\frac{1}{2}$ to $2^{-q \cdot (1-o(1))}$. We shall prove their [Theorem 4.2](#) using a Fourier-based analysis provided in [Håstad and Wigderson](#)

[2003].¹ We point out that reducing the acceptance probability of testers for the Hadamard code and related codes has important implications to PCPs and inapproximability results that are beyond the scope of this course. See e.g., Samorodnitsky and Trevisan [2000, 2006] for more information on this topic.

Definition 4.1 (Graph-tester). Let G be a graph with ℓ vertices numbered $1, \dots, \ell$ and edge set E . For $k \in \mathbb{N}^+$ let T_{G, H_k} be the tester operating as follows when given oracle access to a supposed Hadamard codeword $w \in F_2^{2^k}$:

- Use ℓk random bits to select $x_1, \dots, x_\ell \in F_2^k$.
- Accept iff $T_{H_k}^w[x_i, x_j]$ for all $(i, j) \in E$, where T_{H_k} is the basic BLR-tester from Definition 2.2.

Let $T_{G, H_k}^w[x_1, \dots, x_\ell]$ denote the output of the tester when x_1, \dots, x_ℓ are the random elements selected in the first step.

Theorem 4.2 (Graph-Tester Soundness). *Let G be a graph over ℓ vertices. For all $k \in \mathbb{N}^+$ letting T_G denote the graph-tester T_{G, H_k} ,*

$$\Pr_{x_1, \dots, x_\ell} [T^w[x_1, \dots, x_\ell] = \text{accept}] \leq 2^{-|E|} + \max \left\{ \hat{f}_\alpha \mid \alpha \in F_2^k \right\}.$$

Problem 4.5. Of all ℓ -vertex graphs, which one gives the best bound on the acceptance probability in Theorem 4.2?

4.3 Proof of Theorem 4.2

Problem 4.6. Express the acceptance probability of the graph-tester as a sum of expectations of products of the three-term product $f_w(x_i) \cdot f_w(x_j) \cdot f_w(x_i + x_j)$.

The following Lemma is the crux of the proof of Theorem 4.2.

Lemma 4.3. *For any $S \subseteq E, S \neq \emptyset$,*

$$\mathbf{E}_{x_1, \dots, x_\ell} \left[\prod_{(i, j) \in S} f_w(x_i) \cdot f_w(x_j) \cdot f_w(x_i + x_j) \right] \leq \max \left\{ \hat{f}_\alpha \mid \alpha \in F_2^k \right\}.$$

Problem 4.7. Prove Theorem 4.2 using the previous Lemma 4.3.

For the remainder of this session we focus on proving Lemma 4.3.

Problem 4.8. Assume without loss of generality $(1, 2) \in E$. Argue one can fix values to all random variables but for x_1, x_2 such that the residual expectation, taken only over x_1, x_2 , is at least as large as the left hand side in Lemma 4.3.

¹The work of Håstad and Wigderson also reduces the second summand exponentially in \sqrt{q} and we refer the interested reader to their work for more details.

Problem 4.9. Using the previous problem, show there exist two functions $g, h : F_2^k \rightarrow \{-1, 1\}$ such that

$$\mathbf{E}_{x_1, \dots, x_\ell} \left[\prod_{(i,j) \in S} f_w(x_i) \cdot f_w(x_j) \cdot f_w(x_i + x_j) \right] \leq \mathbf{E}_{x_1, x_2} [g(x_1) \cdot h(x_2) \cdot f_w(x_1 + x_2)].$$

Problem 4.10. Use the Fourier representation of g, h, f_w to argue

$$\mathbf{E}_{x_1, \dots, x_\ell} \left[\prod_{(i,j) \in S} f_w(x_i) \cdot f_w(x_j) \cdot f_w(x_i + x_j) \right] \leq \sum_{\alpha \in F_2^k} \hat{g}_\alpha \hat{h}_\alpha \hat{f}_\alpha.$$

Problem 4.11. Complete the proof of [Lemma 4.3](#) using the previous problem, Parseval's identity and the Cauchy-Schwartz inequality: $\langle g, h \rangle \leq \|g\| \cdot \|h\|$.

In the past sessions we have seen that Hadamard codes are in many respects marvelous locally testable codes. They can be tested by making three queries, each query is answered by a single bit and the resulting soundness function is linear in the distance of the tested word. The main drawback of these codes is their *inverse exponential rate* — we need 2^k bits to encode a message of length k .

In the next few sessions we will show how to obtain LTCs with better rate. These codes will be based on low-degree polynomials. The query complexity will be larger than 3 bits, however it will still be sublinear in the blocklength and dimension of the code. Time permitting, we will also hint on how one can obtain good rate *and* constant query complexity.

5.1 Reed-Solomon codes

Our starting point is the famous family of codes introduced by [Reed and Solomon \[1960\]](#). These codes have many applications in theoretical computer science, coding theory and in practice. E.g., whenever you listen to a CD or watch a DVD, your player is decoding information encoded via Reed-Solomon codes.

Definition 5.1 (Reed-Solomon codes). Let F_q be a finite field with q elements, $S \subseteq F_q$ and $k \leq q$ an integer. The *Reed-Solomon* code of degree less than k , evaluated over S is

$$\text{RS}(q, S, k) = \left\{ p : S \rightarrow F_q \mid \deg(p) < k \right\},$$

where $\deg(p)$ is the minimal degree of a polynomial that agrees with p on S .

When the field F_q is clear from the context, we will omit the reference to q .

Problem 5.1. Prove: $\text{RS}(q, S, k)$ is a linear code over F_q . What are the dimension, blocklength, distance and rate of this code? Devise an encoding for this code, i.e., a mapping $E : F_q^\ell \rightarrow F_q^b$ whose image is $\text{RS}(q, S, k)$, where ℓ, b are the dimension and blocklength, respectively.

Recall that for every set of k points $\{(x_i, y_i)\}_{i=1}^k$ satisfying $x_i \neq x_j$ for $i \neq j$, there exists a unique polynomial of degree less than k such that $p(x_i) = y_i, i = 1, \dots, k$.

Problem 5.2. What is the minimal query complexity of a tester for $\text{RS}(F, S, k)$ that has perfect completeness and a nonzero soundness function?

Problem 5.3. Suggest a tester for $\text{RS}(q, S, k)$ with query complexity $k + 1$ and analyze its soundness.

In typical coding applications $|S| \gg k$. Thus, the query complexity mentioned above is smaller than the blocklength. However, it is always larger than the dimension of the code, i.e., the information read by the tester is greater than the informational content of the codeword. A natural approach to reduce the query complexity below this bound is to

work with codes based on multivariate polynomials, also known as *Reed-Muller codes* Reed [1954]; Muller [1954]. Although these codes can be defined for any number of variables, we start with the simplest case of bivariate polynomials. We shall later argue that if m -variate polynomials are locally testable, then so are m' -variate polynomials for any $m' \geq m$, so without loss of generality, solving the bivariate case is the hardest and most interesting scenario.

5.2 Bivariate low-degree testing

A bivariate polynomial over F_q is a formal expression of the form $P(x, y) = \sum_{i,j=0}^{\infty} a_{ij} \cdot x^i y^j$. where $a_{ij} \in F_q$ and x, y are formal variables. The *degree* of P in x is $\deg_x(P) = \max\{i \mid a_{ij} \neq 0\}$, the degree in y is analogously defined and the *total degree* is $\deg(P) = \max\{i + j \mid a_{ij} \neq 0\}$. If $S \subseteq F_q \times F_q$ and $f : S \rightarrow F_q$, the total degree of f is the minimal total degree of a polynomial that agrees with f on S . The degrees of f in x and in y are similarly defined.

Problem 5.4. Generalize Definition 5.1 and define the family of *bivariate Reed-Muller codes*. What are the basic parameters of these code, i.e., the dimension, blocklength and distance? Are these codes linear?

The key idea that will allow us to test bivariate Reed-Muller codes (as well as an m -variate Reed-Muller codes, for $m \geq 3$) with query complexity that is significantly smaller than the dimension of the code is captured by our next problem.

Definition 5.2 (Rows and columns). For $f : X \times Y \rightarrow F_q$ and $y_0 \in Y$, the y_0 -row column of f is the univariate function $f_{y_0} : X \rightarrow F_q$ given by $f_{y_0}(x) = f(x, y_0)$. The x_0 -column of f , denoted $f_{x_0} : Y \rightarrow F_q$ is analogously defined.

Problem 5.5. What is the ratio between the dimension of bivariate Reed-Muller codes and the dimension of the rows/columns of the same codes? (Bonus: What is this ratio in the case of m -variate Reed-Muller codes?)

Lemma 5.3. Let $X, Y \subseteq F_q$. Suppose $f : X \times Y \rightarrow F_q$ is such that for a μ_y -fraction of rows $y_0 \in Y$ we have $\deg(f_{y_0}) < k$ and similarly for a μ_x -fraction of columns $x_0 \in X$ we have $\deg(f_{x_0}) < k$. Then there exists a bivariate polynomial $P(x, y)$, $\deg_x(P), \deg_y(P) < k$ such that P agrees with f on at least a $(\mu_x + \mu_y - \mu_x \mu_y)$ -fraction of $X \times Y$.

Problem 5.6. Prove Lemma 5.3. Hints: (i) Let Y', X' denote the sets of “good” rows and columns. (ii) Pick k good rows and interpolate² to obtain a bivariate polynomial $P(x, y)$; (iii) Argue $P(x, y)$ agrees with all good rows; (iv) Argue it agrees with all good columns.

²The following fact may come in handy: For any $X'' \subseteq F_q$ and $x_0 \in X''$ there exists a polynomial $P_{x_0}(x)$, $\deg(P_{x_0}) < |X''|$ such that

$$P_{x_0}(x) = \begin{cases} 1 & x = x_0 \\ 0 & x \in X'' \setminus \{x_0\} \end{cases}$$

Problem 5.7. Use the previous problem to construct LTCs based on bivariate Reed-Muller codes that have (i) rate $1/4$, (ii) query complexity roughly square root the code's dimension, (iii) perfect completeness (iv) soundness function $s(\delta) \geq \Omega(\sqrt{\delta})$ where δ denotes the distance of a word from the code. (Bonus question: Construct a tester for *total degree* k .)

In the previous session we proved [Lemma 5.3](#) saying, informally, that if $f : X \times Y \rightarrow F_q$ is far from any low-degree bivariate polynomial, then a random row/column of f has *nonzero distance* from all univariate low-degree polynomials. In this session we'll start proving a stronger statement, namely that a random row/column of f is far from all univariate low-degree polynomials. In other words, large distance of the “big” object f of size n^2 from the “big” code — the set of bivariate polynomials — implies expected large distance of “small” objects f_{x_0}, f_{y_0} of size n from the “small” code — the Reed-Solomon code comprised of low-degree univariate polynomials.

In class we argued that obtaining this stronger notion of testability allows composition of locally testable codes and can be used to obtain LTCs with large rate and small (even constant) query complexity. The role of composition in obtaining such LTCs is beyond the scope of this course. The interested reader is referred to our course notes [[Ben-Sasson, 2005](#), Lecture 8]. Our goal in the next two sessions is to prove the following theorem due to [Polishchuk and Spielman \[1994\]](#).

Theorem 6.1 (Bivariate low-degree testing). *Let $X, Y \subseteq F_q, |X|, |Y| = n, d < n/4$ and $\delta < 1/2$. If*

$$\mathbf{E}_{x_0 \in X} [\Delta(f_{x_0}, \text{RS}(q, Y, d))] < \delta/2 \text{ and } \mathbf{E}_{y_0 \in Y} [\Delta(f_{y_0}, \text{RS}(q, X, d))] < \delta/2,$$

Then there exists a bivariate polynomial $P(x, y), \deg_x(P), \deg_y(P) < d$ such that

$$\mathbf{Pr}_{x_0 \in X, y_0 \in Y} [f(x_0, y_0) \neq P(x_0, y_0)] < 10\delta.$$

Remark. The constants in the original statement of [Theorem 6.1](#) in [Polishchuk and Spielman \[1994\]](#) are somewhat better than stated above. We favor simplicity of proofs over constant optimization.

Problem 6.1. Based on the previous theorem and [Problem 5.3](#) devise a tester with query complexity $d + 1$ for bivariate polynomials of degree $< d$ in x and y . Provide a lower bound for its soundness function.

We'll use the algebraic structure of the ring of polynomials to prove the theorem in three steps: (i) Convert the problem to an algebraic question about *division* of one bivariate polynomial by another bivariate polynomial. (ii) Solve the division problem; This is the crux of the proof and we'll address it next session. (iii) Complete the proof, assuming the solution to part (ii).

6.1 Trading degree for errors

For the rest of this and next session we fix X, Y, n, d, f and δ as in the statement of [Theorem 6.1](#). f is “almost” a low-degree bivariate polynomial, but for a δ -fraction of errors. We

would like to obtain truly low-degree polynomials, with no error. Surprisingly, if we slightly increase the degree we're dealing with, we can remove errors. This idea of removing errors by increasing the degree is behind many efficient decoding algorithms for Reed-Solomon codes, e.g., [Berlekamp and Welch](#); [Berlekamp \[1996\]](#) (see also the exposition in [\[Sudan, 2001, Lecture 10\]](#)).

Problem 6.2. Prove there exist $R(x, y), \deg_x(R) < d$ and $C(x, y), \deg_y(C) < d$ such that

$$\Pr_{X \times Y}[R(x_0, y_0) \neq f(x_0, y_0)] < \delta/2 \text{ and } \Pr_{X \times Y}[C(x_0, y_0) \neq f(x_0, y_0)] < \delta/2$$

Conclude $\Pr_{X \times Y}[R(x, y) \neq C(x, y)] < \delta$.

We call R the *row-polynomial* to denote the fact that each row of R is low-degree. Similarly, C is called the *column-polynomial*. The construction discussed in the next two problems removes noise at the expense of increasing the degree.

Problem 6.3. Prove: For any subset $S \subset X \times Y, |S| < s$, there exists a nonzero polynomial $E(x, y), \deg_x(E), \deg_y(E) < \sqrt{s}$ such that $E(x_0, y_0) = 0$ for all $(x_0, y_0) \in S$. Hint: The set of low-degree bivariate polynomials forms a vector space over F_q .

Problem 6.4. Prove: There exist bivariate polynomials $E(x, y), \deg_x(E), \deg_y(E) < \sqrt{\delta n}$ and $Q(x, y), \deg_x(Q), \deg_y(Q) < d + \sqrt{\delta n}$ such that

$$E(x_0, y_0) \cdot R(x_0, y_0) = E(x_0, y_0) \cdot C(x_0, y_0) = Q(x_0, y_0) \text{ for all } x_0 \in X, y_0 \in Y. \quad (1)$$

From here on we fix R, C, E, Q with degrees as stated above and assume Equation (1) holds.

6.2 Bivariate Division

Problem 6.5. Prove: For every $y_0 \in Y$, we have $E(x, y_0)$ divides $Q(x, y_0)$, i.e., there exists a univariate polynomial $P_{y_0}(x)$ such that $E(x, y_0) \cdot P_{y_0}(x) = Q(x, y_0)$. Similarly, $E(x_0, y)$ divides $Q(x_0, y)$ for every $x_0 \in X$.

The crux of the proof of [Theorem 6.1](#) is that $E(x, y)$ divides $Q(x, y)$ in the ring of bivariate polynomials $F_q[x, y]$, i.e., there exists $P(x, y)$ such that $E(x, y) \cdot P(x, y) = Q(x, y)$. (Notice the equality here is between the polynomials as formal expressions in x, y .) We defer the proof of this statement to the next session.

Lemma 6.2 (Bivariate division). *Let $E(x, y), Q(x, y)$ be bivariate polynomials over F_q with $\deg_x(E) = a \leq \deg_y(E) = b$ and $\deg_x(Q) \leq a + d, \deg_y(Q) \leq b + d$. Let $X, Y \subseteq F_q, |X| = |Y| = n$ where $n > 2(a + d)$. Suppose $E(x, y_0)$ divides $Q(x, y_0)$ for all $y_0 \in Y$ and $E(x_0, y)$ divides $Q(x_0, y)$ for all $x_0 \in X$, then $E(x, y)$ divides $Q(x, y)$ in the ring of bivariate polynomials $F_q[x, y]$, i.e., there exists $P(x, y)$ such that $E(x, y) \cdot P(x, y) = Q(x, y)$.*

We ended the last session by claiming that bivariate division ([Lemma 6.2](#)) implies bivariate low-degree testing ([Theorem 6.1](#)). We start this session by proving this implication and completing the proof of [Theorem 6.1](#). Then we'll start the proof of the bivariate division [Lemma 6.2](#). We'll complete the proof in our next session, which will be the final session on locally testable codes in this course.

7.1 Completing the proof of [Theorem 6.1](#)

To prove [Theorem 6.1](#) we need to find a bivariate polynomial of degree less than d in x and in y and show it agrees with f on most points. We start by suggesting the candidate low-degree polynomial.

Problem 7.1. Prove: [Lemma 6.2](#) implies $\deg_x(P), \deg_y(P) < d$.

Next we show our candidate agrees with f on all but a 10δ -fraction of points in $X \times Y$.

Problem 7.2. Prove: [Lemma 6.2](#) implies $\Pr_{X \times Y}[P(x_0, y_0) \neq R(x_0, y_0)] \leq 9\delta$. Hints:

- Notice $P(x_0, y_0) \cdot E(x_0, y_0) = R(x_0, y_0) \cdot E(x_0, y_0) = C(x_0, y_0) \cdot E(x_0, y_0)$ for all $x_0 \in X, y_0 \in Y$.
- Consider the set of *good* points on which R and C agree. Notice this set is large.
- If P disagrees with R on a *good* point (x_0, y_0) then P must disagree with R on most of the y_0 -row and with most of C on the x_0 -column
- E has a root at every point of disagreement between either pair of P, R, C
- E is low-degree, hence has few roots (see [Lemma 7.1](#)).

The bound on the number of zeros of low-degree multivariate polynomials is given by the following useful Lemma. Its proof is by induction on the number of variables.

Lemma 7.1 (Schwartz-Zippel). *Let $E(x_1, \dots, x_m)$ be a nonzero polynomial of total degree less than d . Let $S = X_1 \times \dots \times X_m, |X_i| = n$. Then the number of zeros of E in S is bounded by $n^{m-1} \cdot d$.*

Problem 7.3. Complete the proof of [Theorem 6.1](#) using the problems and Lemmas in this and the previous session.

7.2 Proving [Lemma 6.2](#)

Suppose we want to prove that 15 divides 300. One way to do this is to prove that 5 is common factor, divide both numbers by 5 and then prove that $3 = 15/5$ divides $60 = 300/5$. We'll take a similar approach in our proof of [Lemma 6.2](#). To prove that $E(x, y)$ divides

$Q(x, y)$ we'll find a *nontrivial common factor*, i.e., a polynomial $P(x, y)$, $\deg(P) > 0$ such that P divides both E and Q . Once the common factor is removed, we'll repeat the process until E becomes a constant. In the rest of this session and our next session we'll prove the following claim.

Lemma 7.2 (Bivariate factoring). *Let $E(x, y), Q(x, y)$ be bivariate polynomials over F_q with $\deg_x(E) = a \leq \deg_y(E) = b$ and $\deg_x(Q) \leq a + d, \deg_y(Q) \leq b + d$. Let $X, Y \subseteq F_q, |X| = |Y| = n$ where $n > 2(a + d)$. Suppose $E(x, y_0)$ divides $Q(x, y_0)$ for all $y_0 \in Y$ and $E(x_0, y)$ divides $Q(x_0, y)$ for all $x_0 \in X$, then $E(x, y)$ and $Q(x, y)$ have a nontrivial common factor in the ring of bivariate polynomials $F_q[x, y]$, i.e., there exists $P(x, y)$, $\deg(P) > 0, E'(x, y), Q'(x, y)$ such that $E'(x, y) \cdot P(x, y) = E(x, y)$ and $Q'(x, y) \cdot P(x, y) = Q(x, y)$.*

Problem 7.4. Prove: **Lemma 7.2** implies **Lemma 6.2**.

In our final session on locally testable codes we'll prove the bivariate factoring [Lemma 7.2](#). Recall [Lemma 7.2](#) implies the bivariate division [Lemma 6.2](#) (see [Problem 7.4](#)) and the latter lemma implies the bivariate low-degree testing [Theorem 6.1](#) (see [section 7.1](#)). Thus, by the end of this session we will have provided a complete proof of [Theorem 6.1](#).

8.1 The Resultant

The reason we choose to prove the bivariate division [Lemma 6.2](#) by repeated removal of common factors, is as follows. Two polynomials share a nontrivial factor iff a certain matrix is not of full rank. Thus, we reduce the problem of finding a common to that of analyzing the determinant of a matrix. We start by showing this for univariate polynomials.

Problem 8.1. Prove: Two univariate polynomials $E(x), Q(x)$ of degree d and e respectively have a nontrivial common factor iff there exist two polynomials $A(x), B(x)$ of degree less than e, d respectively such that $E(x) \cdot A(x) - Q(x) \cdot B(x) = 0$.

Problem 8.2. Construct a $(d + e) \times (d + e)$ -matrix $M(E, Q)$ whose entries come from the set of coefficients of $E(x)$ and $Q(x)$, such that $\det(M(E, Q)) = 0$ if and only if E and Q have a nontrivial common factor.

The determinant of $M(E, Q)$ is called the *resultant* of E and Q and is denoted by $R(E, Q) = \det(M(E, Q))$.

8.2 The Resultant polynomial

Recall we wish to find a common factor of *bivariate* polynomials. To this end, we extend the concept of a resultant to deal with bivariate polynomials. The idea is to construct a matrix whose entries are *polynomials* in x and notice the determinant of this matrix is also a polynomial in x .

Problem 8.3. Construct a $(d + e) \times (d + e)$ -matrix $M(E, Q)(x)$ whose entries are *univariate polynomials in x* , such that for all $x_0 \in F_q$

$$R(E, Q)(x_0) = 0 \Leftrightarrow E(x_0, y) \text{ and } Q(x_0, y) \text{ have a nontrivial common factor,}$$

where $R(E, Q)(x_0) = \det(M(E, Q)(x_0))$ and $M(E, Q)(x_0)$ is obtained by evaluating each entry of $M(E, Q)(x)$ at x_0 .

Problem 8.4. Give an upper bound on $\deg(R(E, Q)(x))$ in terms of $\deg_x(E)$, $\deg_y(E)$, $\deg_x(Q)$ and $\deg_y(Q)$. Conclude that if $E(x_0, y)$ divides $Q(x_0, y)$ for sufficiently many $x_0 \in X$, then $R(E, Q)(x) = 0$, i.e., $R(E, Q)(x)$ is the zero polynomial. How many x_0 's do we need?

8.3 Finding a nontrivial common factor

Problem 8.4 shows that if $E(x_0, y)$ divides $Q(x_0, y)$ for many x_0 , then $R(E, Q)(x) = 0$. We wish to conclude that P, E share a nontrivial common factor, thereby proving **Lemma 7.2**. To this end, we next define the algebraic structure in which such a common factor exists.

The field of rational functions The set of integers $\mathbb{Z} = \{0, 1, -1, 2, -2, \dots\}$ with the standard definition of addition and multiplication forms a *ring* — the ring of integers. This ring is a subset of the *field of rational numbers* \mathbb{Q} whose elements are *ratios* of ring-elements — integers. Similarly, the *ring of polynomials* is a subset of the *field of rational functions* whose elements are ratios of polynomials. In particular, if F is a field let $F[x]$ denote the *ring of polynomials* in variable x with the standard definition of (polynomial) addition and multiplication. A *rational function* in variable x is a ratio $\frac{P(x)}{Q(x)}$ of two polynomials $P(x), Q(x) \in F[x]$. The set of rational functions with the standard definition of addition and multiplication is a *field*, aptly termed the *field of rational functions* and denoted $F(x)$ (not to be confused with $F[x]$ — the *ring of polynomials* in x).

Problem 8.5. Prove: Let $E(x, y), Q(x, y) \in F_q[x, y] \subset F_q(x)[y]$. If $R(E, Q)(x) = 0$, then P, E have a nontrivial common factor in $F_q(x)[y]$ — the ring of polynomials in y with coefficients in the field of rational functions $F_q(x)$.

We can conclude that there exists some $p(y) \in F_q(x)[y], \deg_y(p) > 0$ that is a factor of $E(x, y)$ and of $Q(x, y)$. Alas, the coefficients of p are *rational functions* in x . The great Carl Friedrich Gauss comes to our aid. In what follows, the elements of the ring $F[x]$ are called the *integral points* of the field $F(x)$.

Lemma 8.1 (Gauss' Lemma). *Let $Q(y) \in F(x)[y]$ be a polynomial in y whose coefficients are integral points in $F(x)$, i.e., elements of $F[x]$. If $Q(y)$ has a factor in $F(x)[y]$, then $Q(y)$ has a factor with integral coefficients, i.e., $Q(y)$ has a factor in $F[x, y]$.*

Problem 8.6. Use Gauss' Lemma to deduce that **Problem 8.5** implies $E(x, y)$ and $Q(x, y)$ have a nontrivial common factor in $F_q[x, y]$.

8.4 Reducing the number of resultant roots

We are almost done with the proof of **Lemma 7.2**. The only remaining problem is that to apply **Problem 8.5** we need $|X|, |Y| > \Omega(d^2)$, whereas the statement of **Lemma 7.2** only requires $|X|, |Y| > \Omega(d)$. Our final claim is a lower bound on the *multiplicity* of the roots of $R(P, E)(x)$. Recall that a root $r \in F$ of a polynomial $p(x) \in F[x]$ is said to have *multiplicity* m if $(x - r)^m$ divides $p(x)$ in $F[x]$. We omit the proof of the following claim and refer the interested reader to [Polishchuk and Spielman, 1994, Lemma 8].

Claim 2 (Multiplicity of resultant roots). *Let $R(E, Q)(x)$ be the resultant polynomial as defined in **Problem 8.4** for polynomials E, Q as in **Lemma 7.2**. Then each root $x_0 \in X$ of $R(E, Q)(x)$ has multiplicity at least b .*

Problem 8.7. Complete the proof of **Lemma 7.2** using the previous claim (and problems and lemmas in the past sessions).

8.5 Suggested research projects regarding LTCs

Hadamard codes

- **Randomness complexity of the BLR test:** The standard BLR test requires $2k$ random bits to test the k -dimensional Hadamard code H_k . Can one test the code with less randomness? Related reading: [Ben-Sasson et al. \[2003\]](#); [Shpilka and Wigderson \[2004\]](#).
- **Improved soundness for Hadamard testing:** What is the best asymptotic soundness that one can achieve in testing the Hadamard code and using q query bits? Related reading: [Samorodnitsky and Trevisan \[2000, 2006\]](#).
- **Fourier analysis of general homomorphism testing:** Generalize the Fourier analysis of the BLR tester to the case of arbitrary homomorphism testing. Related reading: [Grigorescu et al. \[2006\]](#).
- **Find a puncturing of the Hadamard code that is locally testable with constant query complexity:** Find a linear locally testable code whose generating matrix is formed by a subset of the rows of the generating matrix of the Hadamard code.

Reed-Solomon and Reed-Muller codes

- **Low degree testing with improved soundness:** Improve the soundness function of the low-degree test. Related reading: [Polishchuk and Spielman \[1994\]](#); [Raz and Safra \[1997\]](#); [Arora and Sudan \[2003\]](#).
- **Generalized low-degree testing for tensor codes:** Find other codes for which a low-degree like test exists. Related reading: [Ben-Sasson and Sudan \[2006\]](#); [Dinur et al. \[2006\]](#)

SESSION 9:
LATTICES — INTRODUCTION

INSTRUCTOR: Eli Ben-Sasson

DECEMBER 31, 2007

For the rest of this course we will study *lattices* in the context of Computer Science. In particular, we will study the so called LLL-algorithm of [Lenstra et al. \[1982\]](#) and some of its applications. Time permitting, we will discuss the current knowledge about the computational complexity of the *closest vector problem*, following the works of [Goldreich and Goldwasser \[2000\]](#) and [Aharonov and Regev \[2005\]](#). Our exposition will closely follow the course notes of [Regev \[2004a\]](#). In particular, in this session we'll cover the first lecture from these notes.

Today we will study the *LLL*³ *basis reduction algorithm* of [Lenstra et al. \[1982\]](#). This algorithm produces a lattice vector whose length is within a $2^{O(n)}$ -factor of the shortest nonzero lattice vector. Although the approximation factor is exponential, we shall see that it suffices to achieve several interesting corollaries, such as breaking certain variants of RSA cryptosystems. We will follow [[Regev, 2004a](#), Lecture 2] in our exposition of the algorithm and its analysis. In particular, we will rely on definitions and claims from Lecture 1 in Regev’s course notes.

10.1 Reduced basis

Recall the Gram-Schmidt orthogonal basis.

Definition 10.1 (Gram-Schmidt Basis). Given linearly independent $b_1, \dots, b_n \in \mathbb{R}^n$, their *Gram-Schmidt orthogonal* basis is given by

$$\tilde{b}_i = b_i - \sum_{j < i} \mu_{i,j} \tilde{b}_j, \quad \text{for } \mu_{i,j} = \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}.$$

Problem 10.1. Prove: If b_1, \dots, b_n are orthogonal, then a shortest vector of $\mathcal{L}(B)$ can be found efficiently.

One way to view a reduced basis, defined next, is as a “somewhat-orthogonal” basis. Such a basis facilitates finding a “somewhat-short” nonzero lattice vector.

Definition 10.2 (Reduced basis). A basis $B = \{b_1, \dots, b_n\} \in \mathbb{R}^n$ is said to be a δ -*LLL reduced basis* if the following two properties hold.

1. $|\mu_{i,j}| \leq 1/2$ for all $1 \leq i \leq n, j < i$.
2. $\|\tilde{b}_{i+1}\|^2 \geq (\delta - \mu_{i+1,i}^2) \|\tilde{b}_i\|^2$ for all $1 \leq i \leq n$.

Informally, the first property says b_1, \dots, b_n are somewhat orthogonal — complete orthogonality would give $\mu_{i,j} = 0$; The second property says that $\tilde{b}_1, \dots, \tilde{b}_n$ are somewhat long. To see this set $\delta = 3/4$ and notice property 1 implies $\|\tilde{b}_i\|^2 \geq \|\tilde{b}_{i-1}\|^2/2$. The focus of our session today is the following fundamental theorem of [Lenstra et al. \[1982\]](#).

Theorem 10.3 (Efficient basis reduction). *For any fixed $\delta \in (\frac{1}{4}, 1)$ the following holds. Any lattice basis b_1, \dots, b_n can be converted to a δ -LLL reduced basis in polynomial time in the description of b_1, \dots, b_n .*

³The term LLL is an abbreviation of the three inventors of the algorithm and coauthors of [Lenstra et al. \[1982\]](#) — A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovasz.

Our next problem shows one important application of a reduced basis.

Problem 10.2. Present a polynomial-time algorithm that given a basis $B = b_1, \dots, b_n$ finds a nonzero lattice vector of length $\leq 2^{n/2} \cdot \lambda_1(\mathcal{L}(B))$, where $\lambda_1(\mathcal{L})$ is the length of a shortest nonzero vector in the lattice \mathcal{L} . Hints:

- Use **Theorem 10.3**. Assume b_1, \dots, b_n is δ -LLL reduced and let $\tilde{b}_1, \dots, \tilde{b}_n$ be the corresponding Gram-Schmidt orthogonal basis.
- Recall [Regev, 2004a, Lecture 1, Theorem 5]: $\lambda_1(\mathcal{L}(B)) \geq \min_i \|\tilde{b}_i\|$.
- Using properties 1,2 in **Definition 10.2** prove $\tilde{b}_1 = b_1$ is not much longer than $\min_i \|\tilde{b}_i\|$.

10.2 The LLL Algorithm

Theorem 10.3 is proven by analyzing the correctness and running time of the following algorithm, presented in Lenstra et al. [1982].

The LLL algorithm

Input: Lattice basis $B = b_1, \dots, b_n \in \mathbb{Z}^n$, $\delta \in (1/4, 1)$

Start: Compute Gram-Schmidt orthogonal basis $\tilde{b}_1, \dots, \tilde{b}_n$

Reduction: For $i = 2$ to n do

For $j = i - 1$ down to 1 do

$$b_i \leftarrow b_i - c_{i,j} b_j, \text{ for } c_{i,j} \text{ the integer closest to } \frac{\langle b_i, \tilde{b}_j \rangle}{\langle \tilde{b}_j, \tilde{b}_j \rangle}$$

Swap: If exists i , $\|\tilde{b}_{i+1}\|^2 < \left(\delta - \mu_{i+1,i}^2\right) \|\tilde{b}_i\|^2$, then

swap b_i and b_{i+1}

goto **Start**

Output: “ b_1, \dots, b_n is a δ -LLL reduced basis for the input lattice.”

Problem 10.3. Prove: The Gram-Schmidt orthonormal basis does not change during the reduction step. (It may change, however, during the swap step.)

Problem 10.4. Prove correctness: If the algorithm terminates, then its output is indeed a δ -LLL reduced basis for the input lattice. Hints: Unimodular matrices and [Regev, 2004a, Lecture 1, Lemma 3].

10.3 Bounding the runtime of the algorithm

We want to show (a) the number of restarts is small, and (b) each iteration is efficient. Combined, these two statements prove the efficiency of the LLL algorithm. To prove (a), we associate a *potential* with each intermediate basis in the algorithm and show (i) an upper bound on the initial potential (ii) a lower bound on the final potential, and (iii) each restart decreases the potential.

Definition 10.4 (LLL Potential). For B a lattice basis, the *potential* of B is

$$D_B = \prod_{i=1}^n \|\tilde{b}_i\|^{n-i-1} = \prod_{i=1}^n D_{B,i},$$

where $D_{B,i} = \prod_{j=1}^i \|\tilde{b}_j\|$.

Problem 10.5. Prove $\log(D_B)$ is at most polynomial in the size of the description of the input basis.

Problem 10.6. Prove D_B decreases by a multiplicative factor $\sqrt{\delta}$ with each restart. Hints:

- Using [Problem 10.3](#) argue that it is sufficient to consider only swap steps.
- Suppose b_i is swapped with b_{i+1} . Argue that no change occurs in $D_{B,j}$ for $j \neq i$. Thus, it is sufficient to consider only the change occurring in $D_{B,i}$.
- Consider the ratio of $D_{B,i}$ before and after the swap.

Problem 10.7. Using the previous two problems, prove that for any $\delta \in (\frac{1}{4}, 1)$ the number of restarts of the LLL algorithm is at most polynomial in the description of the input lattice basis.

We have shown a polynomial upper bound on the number of restarts of the algorithm. However, the calculations involved in a single iteration might require superpolynomial time. For example, the intermediate basis generated by our algorithm might be doubling in length with each restart. The second part of the running time analysis shows this is not the case and all involved calculations are polynomial in the size of the input basis. For a detailed analysis we refer to [\[Regev, 2004a, Lecture 2\]](#).

With this we conclude the description and analysis of the LLL algorithm. In our next session we shall study some of the many applications of this marvelous algorithm.

In this session we'll study one out of many applications of the LLL algorithm. In particular, we will use this algorithm to mount a special kind of attack on the well-known RSA cryptosystem introduced in Rivest et al. [1978]. We start by briefly describing the essentials of this system.

RSA Public key cryptography Let p, q be two primes, $N = p \cdot q$ and $\psi(N) = (p - 1) \cdot (q - 1)$ be the size of \mathbb{Z}_N^* — the multiplicative group of integers modulo N . Let r, s be integers such that $r \cdot s = 1 \pmod{\psi(N)}$. We call r the *public exponent*, s is the *private exponent* and N is the *modulus* of the cryptosystem. The pair (r, N) is the *public key* and the pair (s, N) is the *private key*. A message is $M \in \mathbb{Z}_N^*$. The *encryption* of M is given by $C = M^r \pmod{N}$ and C is called a *cyphertext*. The *decryption* of a cyphertext C is given by $C^s \pmod{N}$. Recall Euler's Theorem says that for all integers a we have $a^{\psi(N)} = 1 \pmod{N}$. This Theorem implies $(M^r)^s = M \pmod{N}$. In other words, the decryption of a cyphertext C recovers the message M encoded by C .

We shall make extensive use of the following fundamental theorem.

Theorem 11.1 (Chinese Remainder). *Given r modular equalities $x = a_i \pmod{m_i}$ where m_1, \dots, m_r are relatively prime, there exists a unique $x < \prod_i m_i$ satisfying these equalities and x can be found in polynomial time in $\log(\prod_i m_i)$.*

11.1 Low Exponent Attack

Since the encryption of messages is often carried out by simple computational devices, like a “smartcard”, we would like to make the encryption process as efficient as possible. One way of doing this would be to use a small public exponent r . In class we showed that using $r = 3$ makes the RSA system susceptible to attack. We now extend this idea significantly, following the works of Coppersmith [1997] and Håstad [1988]. For more information on various attacks on the RSA system (including the one we discuss below), see the survey Boneh [1999].

Suppose Alice goes on an online shopping spree and purchases items from k different online stores. In all transactions, Alice transmits her (unique) credit card number M , using k different RSA public keys $(r_1, N_1), \dots, (r_k, N_k)$. A special case of the following theorem due to Håstad [1988] says that if Eve eavesdrops to all transactions and additionally all public exponents r_1, \dots, r_k are smaller than $\approx \sqrt{k}$, then Eve can go on her own shopping spree, using Alice's credit card number.

Theorem 11.2 (Low exponent attack). *For every integer d there exists an integer $N_0 = N_0(d)$ such that the following holds. Let $k > d(d + 1)$, let $N_1 < \dots < N_k \in \mathbb{N}^+$ be integers all larger than N_0 and let $g_i \in \mathbb{Z}_{N_i}[x]$ be k polynomials of maximum degree d . Suppose there exists a unique $M < N_1$ such that $g_i(M) = C_i \pmod{N_i}$ for $i = 1, \dots, k$. Then one can find M given $S = \{(N_i, g_i, C_i)\}_{i=1}^k$ in time polynomial in the description of the input S .*

Remark. In the theorem above, the minimal number of different RSA-systems needed to break degree d encryption is $\approx d^2$. However, a better bound is known, where one needs only $k > d$. For details, see [Regev, 2004a, Lecture 4] or Boneh [1999].

The proof of **Theorem 11.2** reduces the problem of finding a common root of k low-degree polynomials $g_1(x), \dots, g_k(x)$ modulo k different integers N_1, \dots, N_k , to that of finding a single root of a low-degree polynomial modulo $N_1 \cdots N_k$. We will find such a root via the LLL algorithm, thus proving the following theorem of Coppersmith [1997].

Theorem 11.3 (Finding small modular roots). *For every integer d there exists a constant $c_d > 0$ such that the following holds. Let $N \in \mathbb{N}^+$ and $f \in \mathbb{Z}_N[x]$ be a monic⁴ polynomial of degree d . Let $B = c_d \cdot N^{\frac{2}{d+1}}$. Then one can find the set of small integer roots modulo N ,*

$$R = \{x_0 \in \mathbb{Z} \cap [-B, B] \mid f(x_0) = 0 \pmod{N}\},$$

in time that is polynomial in $\log(N)$.

Remark. As in the case of **Theorem 11.2**, the bound on B stated above is weaker than the best known. We state this weaker version because it is simpler to prove. For details on the stronger versions, see Coppersmith [1997] or Boneh [1999]; Regev [2004a].

11.2 Reducing **Theorem 11.2** to **Theorem 11.3**

We start by assuming without loss of generality that N_i, N_j are *relatively prime*, i.e., their greatest common divisor is 1. Recall an *extended gcd pair* of $a, b \in \mathbb{N}^+$ is a pair $s, t \in \mathbb{Z}$ such that $as + bt = g$ where g is the greatest common divisor of a and b . Recall that an extended gcd pair can be computed in polynomial time in $\log(a) + \log(b)$.

Problem 11.1. Prove: If N_i is not relatively prime to N_j for some $i \neq j$, then one can find M in polynomial time.

The reduction of **Theorem 11.2** to **Theorem 11.3** follows by “combining” all k modular equations into one big equation of the same degree, using the Chinese Remainder **Theorem 11.1**.

Problem 11.2. Prove **Theorem 11.3** implies **Theorem 11.2**. Hints:

- Let $h_i(x) = g_i(x) - C_i$.
- Let $h_i^{(\max)}$ be the leading coefficient of h_i . Use the proof of **Problem 11.1** to argue that without loss of generality $h_i^{(\max)}$ is relatively prime to N_i .
- Using the extended-gcd of $h_i^{(\max)}$ and N_i convert $h_i(x)$ into a *monic* polynomial $h'_i(x)$ of degree *exactly* d such that $h'_i(M) = 0 \pmod{N_i}$.
- Use **Theorem 11.1** to efficiently find integers T_1, \dots, T_k such that $T_i = 1 \pmod{N_i}$ and $T_i = 0 \pmod{N_j}$ for all $j \neq i$. Notice $\sum_i T_i = 1 \pmod{N_1 \cdots N_k}$.
- Argue $h(x) = \sum_{i=1}^k T_i \cdot h'_i(x)$ satisfies the conditions of **Theorem 11.3**, for a proper setting of N .
- Use **Theorem 11.3** to find M efficiently.

⁴A polynomial is said to be *monic* if its highest nonzero coefficient is 1.

11.3 Proof of **Theorem 11.3**

From here on we fix d, N, B and $f(x) = \sum_{i=0}^d a_i x^i$ as in the statement of **Theorem 11.3**. Our proof idea is to find a polynomial with *small coefficients* a'_0, \dots, a'_d that has the same modular roots as $f(x)$. We start by seeing why small coefficients help prove **Theorem 11.3**.

Problem 11.3. Prove: If the coefficients of $f(x)$ satisfy $|a_i B^i| < N/(d+1)$ for all $i = 0, \dots, d$, then the set R can be found efficiently.

The problem is that the coefficients of $f(x)$ need not be small. Thus, we shall find a polynomial $g(x)$ with the same set of modular roots as f and small coefficients. For this, we'll need the LLL algorithm.

Problem 11.4. Consider the set of polynomials $\{N, Nx, Nx^2, \dots, Nx^{d-1}, f(x)\}$. Let $g(x) = \sum_{i=0}^{d-1} \beta_i N x^i + \alpha f(x)$ be an integer combination of these polynomials for $\alpha, \beta_0, \dots, \beta_{d-1} \in \mathbb{Z}$. Prove: Every root of f modulo N is also a root of g modulo N . Moreover, if $\alpha \neq 0$ then the sets of roots of f and g modulo N are equivalent.

Problem 11.5. Consider the lattice \mathcal{L}_1 generated by the columns of the following matrix

$$\begin{pmatrix} N & 0 & \dots & 0 & a_0 \\ 0 & BN & \dots & 0 & a_1 B \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & B^{d-1} N & a_{d-1} B^{d-1} \\ 0 & 0 & \dots & 0 & B^d \end{pmatrix}.$$

Suppose $\ell \in \mathcal{L}_1 \setminus \{0\}$ satisfies $\|\ell\| < \frac{N}{d+1}$. Construct from ℓ a nonzero polynomial with the same set of roots as $f(x)$ that satisfies the condition of **Problem 11.3**. Conclude that finding ℓ suffices to prove **Theorem 11.3**.

Problem 11.6. Provide an upper bound on the length of the shortest nonzero lattice vector of \mathcal{L}_1 . Hints: calculate $\det \mathcal{L}_1$ and use Minkowski's First Theorem (cited as Corollary 2 in [Regev, 2004a, Lecture 1]).

Problem 11.7. Use the LLL algorithm, **Theorem 10.3** and **Problem 11.6** to argue that a nonzero lattice vector of length $c'_d \cdot N \cdot \frac{B^{d/2}}{N^{1/d+1}}$ can be found efficiently, where c'_d is a constant depending only on d .

Problem 11.8. Complete the proof of **Theorem 11.3** using Problems 11.6 and 11.7.

In the past couple of sessions we studied the LLL algorithm which produces lattice vectors of length that is within a $2^{O(n)}$ -factor of the length of the shortest nonzero lattice point. Additionally, we saw an application of this algorithm to cryptanalysis of the RSA system. Today we briefly investigate the computational complexity of two basic problems related to lattices — the “closest vector” and “shortest vector” problems. In particular, we shall argue that the decision version of these problems is NP-complete and discuss the complexity of their approximation versions.

12.1 Decisional CVP is NP-complete

Definition 12.1 (Decisional CVP). The *decisional version* of the *closest vector problem* is

- **Given:** Lattice basis $B \in \mathbb{Z}^{m \times n}$, target vector $t \in \mathbb{Z}^m$ and radius $r \in \mathbb{Q}$.
- **Output:** “YES” if there exists a lattice vector $u \in \mathcal{L}(B)$ such that $\|t - u\| \leq r$. Otherwise, output “NO”.

Problem 12.1. Prove: Decisional CVP is in **NP**.

To prove the **NP**-completeness of decisional CVP, we apply a reduction from the **NP**-complete *subset-sum* problem:

- **Given:** Summands $a_1, \dots, a_n \in \mathbb{Z}$ and sum $S \in \mathbb{Z}$.
- **Output:** “YES” if there exists a subset of the summands that sums to S . Otherwise, output “NO”.

Problem 12.2. Prove: Decisional CVP is **NP**-complete. Hint: Reduce from subset-sum and consider the instance of decisional CVP given by:

$$B = \begin{pmatrix} a_1 & a_2 & \dots & a_n \\ 2 & 0 & \dots & 0 \\ 0 & 2 & \dots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 2 \end{pmatrix} \in \mathbb{Z}^{(n+1) \times n}, \quad t = \begin{pmatrix} S \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{Z}^{n+1}, \quad r = \sqrt{n}.$$

Ajtai [1998] showed that the decisional *shortest vector problem* (SVP), described below, is **NP**-complete under randomized reductions. (The intricate proof of this statement is beyond the scope of our course.) An SVP instance is a pair (B, r) where B is a lattice basis and r is a radius parameter. We are asked to output “YES” if and only if the length of the shortest nonzero lattice vector in $\mathcal{L}(B)$ is at most r .

12.2 The complexity of approximate SVP

Given [Problem 12.2](#), it currently seems hopeless to find an *exact* optimal solution to the CVP problem. The same holds for the decisional SVP problem, as shown by [Ajtai \[1998\]](#). But perhaps one can hope for efficient *approximate* solutions to SVP? In fact, the LLL algorithm does provide an efficient $2^{O(n)}$ -approximation to the SVP problem. We start with a formal definition of a decisional version of the approximate SVP problem.

Definition 12.2 (GAPSVP). For $\gamma : \mathbb{N}^+ \rightarrow [1, \infty)$, let $\text{GAPSVP}_{\gamma(n)}$ be the following promise problem over instances (B, r) where $B \in \mathbb{Z}^{m \times n}$ is a lattice basis and $r \in \mathbb{Q}$ is a radius parameter.

- **YES:** Instances (B, r) such that $\lambda_1(\mathcal{L}(B)) < r$.
- **NO:** Instances (B, r) such that $\lambda_1(\mathcal{L}(B)) \geq \gamma(n) \cdot r$.

Our goal is to show that GAPSVP_n is unlikely to be **NP**-complete. To do so, we use the following Theorem of [Lagarias et al. \[1990\]](#).

Theorem 12.3. *Every lattice \mathcal{L} has a basis b_1, \dots, b_n such that its Gram-Schmidt orthogonal basis $\tilde{b}_1, \dots, \tilde{b}_n$ satisfies*

$$\min \left\{ \|\tilde{b}_1\|, \dots, \|\tilde{b}_n\| \right\} \geq \frac{\lambda_1(\mathcal{L})}{n}.$$

Time permitting, we will prove this theorem in class using properties of *dual lattices*. (See also [\[Regev, 2004a, Lecture 8\]](#)).

Problem 12.3. Prove: $\text{GAPSVP}_n \in \text{coNP}$. Hints: Use [Theorem 12.3](#) and the second bullet in [Problem 10.2](#).

Notice that the previous problem implies that if GAPSVP_n is **NP**-complete, then **NP** = **coNP**. Common belief is that **NP** \neq **coNP** and this belief implies that GAPSVP_n is not (believed to be) **NP**-complete.

12.3 Complexity of SVP — State of the art

Let us highlight several results regarding the tractability of the GAPSVP_γ problem, for different approximation factors γ . [Lenstra et al. \[1982\]](#) showed that for $\gamma = 2^{O(n)}$ the problem lies in **P**. [Schnorr \[1988\]](#) improved this result to show that for $\gamma = 2^{O(n(\log \log n)^2 / \log(n))}$ still lies in **P** and [Ajtai et al. \[2001\]](#) showed that a $\gamma = 2^{O(n \log \log n / \log(n))}$ can be found in randomized polynomial time.

In this session we have seen that for $\gamma = n$ the problem lies in **NP** \cap **coNP**. [Aharonov and Regev \[2005\]](#) have shown that this remains the case even for $\gamma = \sqrt{n}$. Prior to that, [Goldreich and Goldwasser \[2000\]](#) showed that for $\gamma = \sqrt{n / \log(n)}$ the problem lies in **NP** \cap **coAM**. We briefly remark that this latter result is sufficient to imply that $\text{GAPSVP}_{\sqrt{n / \log(n)}}$ is unlikely to be **NP**-complete.

Starting from $\gamma = 1$, [Ajtai \[1998\]](#) showed that GAPSVP_1 is **NP**-complete under randomized reductions. [Micciancio \[2000\]](#) showed that the problem remains **NP**-hard for $\gamma = \sqrt{2}$ under

randomized reductions. Recently, [Khot \[2005\]](#) improved the hardness result to any constant approximation factor. (He also showed super-constant, though sub-polynomial, hardness results based on stronger computational assumptions).

We end this brief survey by pointing out that cryptographic constructions based on lattice problems (e.g., the constructions of [Ajtai and Dwork \[1997\]](#); [Regev \[2004b\]](#)) typically assume that $\text{GAPSVP}_{n^{O(1)}}$ cannot be solved in randomized polynomial time.

References

- Dorit Aharonov and Oded Regev. Lattice problems in NP intersect coNP. *Journal of the ACM*, 52(5):749–765, 2005. Preliminary version in FOCS'04.
- Miklós Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *STOC*, pages 10–19, 1998. URL <http://doi.acm.org/10.1145/276698.276705>.
- Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In ACM, editor, *Proceedings of the twenty-ninth annual ACM Symposium on the Theory of Computing: El Paso, Texas, May 4–6, 1997*, pages 284–293, pub-ACM:adr, 1997. ACM Press. ISBN 0-89791-888-6. URL <http://www.acm.org/pubs/citations/proceedings/stoc/258533/p284-ajtai/>; <http://www.acm.org/pubs/articles/proceedings/stoc/258533/p284-ajtai/p284-ajtai.pdf>.
- Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In ACM, editor, *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing: Hersonissos, Crete, Greece, July 6–8, 2001*, pages 601–610, pub-ACM:adr, 2001. ACM Press. ISBN 1-58113-349-9.
- Sanjeev Arora. *Probabilistic checking of proofs and hardness of approximation problems*. PhD thesis, Princeton University, 1994.
- Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003. URL <http://www.springerlink.com/index/10.1007/s00493-003-0025-0>.
- Babai, Fortnow, Levin, and Szegedy. Checking computations in polylogarithmic time. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 1991.
- Mihir Bellare, Don Coppersmith, Johan Håstad, Marcos A. Kiwi, and Madhu Sudan. Linearity testing in characteristic two. *IEEE Transactions on Information Theory*, 42(6):1781–1795, 1996.
- Eli Ben-Sasson. From error correcting codes to inapproximability via the probabilistically checkable proofs. Course notes, 2005. URL http://www.cs.technion.ac.il/~eli/courses/PCP_Fall_2005/.
- Eli Ben-Sasson and Madhu Sudan. Robust locally testable codes and products of codes. *Random Struct. Algorithms*, 28(4):387–402, 2006. URL <http://dx.doi.org/10.1002/rsa.20120>.
- Eli Ben-Sasson, Madhu Sudan, Salil P. Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *STOC*, pages 612–621. ACM, 2003. ISBN 1-58113-674-9. URL <http://doi.acm.org/10.1145/780542.780631>.
- E. R. Berlekamp. *Algebraic Coding Theory, revised 1984 edition*. Aegean Park Press, Laguna Hills, Ca., 1984.
- E. R. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent Number 4,633,470.
- Elwyn R. Berlekamp. Bounded distance+1 soft-decision reed-solomon decoding. *IEEE Transactions on Information Theory*, 42(3):704–720, 1996.
- Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, December 1993.

- Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999. URL <http://crypto.stanford.edu/~dabo/papers/RSA-survey.ps>.
- Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, Autumn 1997.
- Irit Dinur, Madhu Sudan, and Avi Wigderson. Robust local testability of tensor products of LDPC codes. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 304–315. Springer, 2006. ISBN 3-540-38044-2. URL http://dx.doi.org/10.1007/11830924_29.
- K. Friedl and M. Sudan. Some improvements to total degree tests. In *3rd Israel Symposium on Theory of Computing and Systems*, Natanya, Israel, 1995. IEEE Comp. Soc. Press.
- Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *J. Comput. Syst. Sci.*, 60(3):540–563, 2000.
- Elena Grigorescu, Swastik Kopparty, and Madhu Sudan. Local decoding and testing for homomorphisms. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 375–385. Springer, 2006. ISBN 3-540-38044-2. URL http://dx.doi.org/10.1007/11830924_35.
- Johan Håstad. Solving simultaneous modular equations of low degree. *SIAM Journal on Computing*, 17(2):336–341, April 1988. ISSN 0097-5397 (print), 1095-7111 (electronic). Special issue on cryptography.
- Johan Håstad and Avi Wigderson. Simple analysis of graph tests for linearity and PCP. *Random Struct. Algorithms*, 22(2):139–160, 2003. URL <http://dx.doi.org/10.1002/rsa.10068>.
- Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM*, 52(5):789–808, September 2005. ISSN 0004-5411.
- J. C. Lagarias, Jr. Lenstra, H.W., and C. P. Schnorr. Korkin-zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10:333–348, 1990.
- A. K. Lenstra, H. W. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.
- Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, second edition edition, January 1997.
- F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, Holland, 1977.
- Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035, 2000. URL <http://epubs.siam.org/sam-bin/dbq/article/37303>.
- David E. Muller. Application of boolean algebra to switching circuit design and to error detection. *IRE Transactions on Electronic Computation*, EC-3:6–12, September 1954.
- Alexander Polishchuk and Daniel A. Spielman. Nearly-linear size holographic proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing, STOC'94 (Montréal, Québec, Canada, May 23-25, 1994)*, pages 194–203, New York, 1994. ACM Press.

- Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC*, pages 475–484, 1997. URL <http://doi.acm.org/10.1145/258533.258641>.
- I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, June 1960. ISSN 0368-4245.
- Irving S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *IRE Transactions on Information Theory*, PGIT-4:38–49, September 1954.
- Oded Regev. Lattices in computer science. Course notes, 2004a. URL http://www.cs.tau.ac.il/~odedr/teaching/lattices_fall_2004/index.html.
- Oded Regev. New lattice-based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, November 2004b. ISSN 0004-5411.
- R. L. Rivest, A. Shamir, and L. Adelman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.
- Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *STOC*, pages 191–199, 2000. URL <http://doi.acm.org/10.1145/335305.335329>.
- Alex Samorodnitsky and Luca Trevisan. Gowers uniformity, influence of variables, and PCPs. In Jon M. Kleinberg, editor, *STOC*, pages 11–20. ACM, 2006. ISBN 1-59593-134-1. URL <http://doi.acm.org/10.1145/1132516.1132519>.
- C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *Journal of Algorithms*, 9(1):47–62, March 1988.
- Jean Pierre Serre. *Linear representations of finite groups*, volume 42 of *Graduate texts in Mathematics*. Springer, 1977.
- Amir Shpilka and Avi Wigderson. Derandomizing homomorphism testing in general groups. In László Babai, editor, *STOC*, pages 427–435. ACM, 2004. ISBN 1-58113-852-0. URL <http://doi.acm.org/10.1145/1007352.1007421>.
- D. A. Spielman. *Computationally Efficient Error-Correcting Codes and Holographic Proofs*. Technical report, June 1995. URL <ftp://ftp-pubs.lcs.mit.edu/pub/lcs-pubs/tr.outbox/MIT-LCS-TR-662.ps.gz>.
- Madhu Sudan. Algorithmic introduction to coding theory. Course notes, 2001. URL <http://theory.lcs.mit.edu/~madhu/FT01/>.
- Luca Trevisan. Recycling queries in PCPs and in linearity tests (extended abstract). In ACM, editor, *Proceedings of the thirtieth annual ACM Symposium on Theory of Computing: Dallas, Texas, May 23–26, 1998*, pages 299–308, pub-ACM:adr, 1998. ACM Press. ISBN 0-89791-962-9. URL <http://www.acm.org/pubs/citations/proceedings/stoc/276698/p299-trevisan/>; <http://www.acm.org/pubs/articles/proceedings/stoc/276698/p299-trevisan/p299-trevisan.pdf>.