6.897 Algorithmic Introduction to Coding Theory	October 31, 2001
Lecture 13	
Lecturer: Madhu Sudan	*Scribe: Hoeteck Wee*

Today's topics:

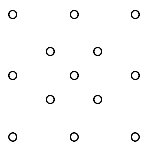
- Review of & improvement to list-decoding from last lecture.
- Mo' better list-decoding.
- Towards generalization of decoding algorithms.

1 Review of List-Decoding

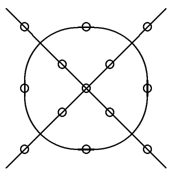
We will start by reviewing the list-decoding algorithm for Reed-Solomon codes from the last lecture, from a very high-level (or hand-wavy, if you prefer) perspective. The goal is to stress the simplicity of the algorithm. Then we will do a careful analysis of the algorithm to see how to get rid of one of the two $\sqrt{2}$ factors that separate the algorithm from the combinatorial bounds.

1.1 The algorithm on an example

To recall the idea of the algorithm, it is illustrative to consider an example. We will consider an example consisting of 13 points in the real (Euclidean) plane $(x_i, y_i), 1 \leq i \leq 13$, We will try to find all degree one polynomials passing through at least five points. Note that the numbers are chosen for illustration only. Certainly the ability to find degree one polynomials should not be considered the high-point of the algorithm, since "brute-force" algorithms could do this in quadratic time.



The main notion that distinguishes this approach from the traditional coding-theoretic approach is that instead of pulling out our calculators and interpolating, we are sitting back and contemplating. We try to find a nice algebraic curve that can explain all the 13 points. One such curve is drawn below.



The curve turn out to be $Q(x, y) = x^4 - y^4 - x^2 + y^2 = 0$ (assuming the central point is the origin and that the points are scaled appropriately). The picture above is the plot of all its zeroes. Clearly there are three distinct components to the set of all zeroes. These correspond to the factorization of Q

whose factors are $x-y$, $x+y$ and x^2+y^2-1. While the algebraic operation of factorization is needed to tell the components, the human eye doesn't need this calculation either - since the picture clearly distinguishes the three factors. In fact a close look at existing factorization algorithms for multivariate polynomials reveals that they are based on the geometric intuition that is evident in the picture above — they factors can be derived by fitting polynomials to zome zero of $Q(x,y)$ and its local naighborhood.

1.2 Weighted degree and some minor improvements

To get the best possible performance out of the algorithm above, we need to be more careful with the degree of the polynomial Q that we pick. For example, when we are trying to find a list of all degree 1 polynomials, p, it is better to pick Q of the smallest possible total degree, rather than the smallest individual degrees in x and y. Based on this idea, and keeping in mind that at some point we end up substituting for y a degree k polynomial p in x (and that we have no idea what this polynomial p is going to be), it is best to fit a polynomial Q that minimizes a "weighted" degree function. We define this weighted degree next:

Definition 1 *The* (k_1,k_2)-weighted degree *of a monomial x^iy^j is the quantity $i\cdot k_1 + j\cdot k_2$. The* (k_1,k_2)-weighted degree *of a polynomial $Q(x,y)$ is the maximum, over all monomials in Q with non-zero coefficients, of their (k_1,k_2)-weighted degree.*

We now present the fully optimized algorithm for Reed-Solomon decoding:

RS-list-decoder-2:

Given Set of n distinct pairs $\{(x_i,y_i)|i\in[n]\}$, and integers t and k.

Parameter: D to be set later.

Step 1: Find a bivariate polynomial $Q(x,y)$ of $(1,k)$-weighted degree D such that:

- $Q(x_i,y_i)=0 \quad \forall i.$
- $Q\not\equiv 0$.

Step 2: Factor $Q(x,y)$ and output the list of polynomials p of degree at most k provided:

- $y-p(x)\mid Q(x,y)$, and
- $p(x_i)=y_i$ for at least t values of $i\in[n]$.

The standard claims adapted for the case of weighted degrees are given below. We skip the proofs, but they are obvious.

Proposition 2 *For any set $S=\{(x_i,y_i)|i\in[n]\}$ of n points in $\mathbb{F}_q\times\mathbb{F}_q$, there exists a non-zero polynomial Q of $(1,k)$-weighted degree at most $\sqrt{2nk}$ such that $Q(x_i,y_i)=0$ for all $i\in[n]$.*

Lemma 3 *If a polynomial $Q(x,y)$ of $(1,k)$-weighted degree D and the polynomial $y-p(x)$, with p having degree at most k, have more than D common zeroes, then $y-p(x)$ divides $Q(x,y)$.*

Putting the two together, we get:

Theorem 4 ([13]) *The algorithm* **RS-list-decoder-2** *with the setting $D=\sqrt{2nk}$ solves the* **RS list decoding problem** *in polynomial time provided $t>\sqrt{2kn}$.*

We now move towards a better algorithm for list-decoding. But first we try to motivate it. Why should we try to improve this algorithm? One of the main motivations is the goal to get the combinatorial results and algorithmic bounds to match. The algorithm above ends up solving the list-decoding problem exactly when the inclusion-exclusion bound works. What we want next is an algorithm that meets the graph-theoretic bound, i.e., when $t>\sqrt{kn}$.[1].

[1]the case $t\le\sqrt{kn}$ is not known

2 An improved list-decoder for RS codes

Before proceeding let us be warned that the "improvement" is only in the number of errors it can correct; and not in the running time. Our running times will actually get worse, but remain polynomially bounded.

2.1 Weighted Decoding Problem

We will actually find an indirect route to our improvement. Rather than solving the standard list-decoding problem, we will explore a weighted list-decoding algorithm.

> **Weighted Reed Solomon List-Decoding Problem:**
> GIVEN: n distinct points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ with corresponding integer weights $w_1, w_2, \ldots, w_n$ (specified in unary), degree bound k, and an agreement bound W (which replaces the parameter t in the previous problem statement).
> GOAL Find all degree k polynomials p such that $\sum_{i|p(x_i)=y_i} w_i > W$

To motivate this problem, let us note that this is in the spirit of the GMD (generalized minimum distance) decoding problem of Forney [7]. The main differences being that (1) Forney's problem associated a "penalty" with each pair (x_i, y_i) and the goal was to find polynomials that minimized the total penalties, and (2) Forney's problem expected the x_i's to be distinct and we won't. As in the case of the GMD algorithm of Forney, a solution to the weighted RS list-decoding problem has the potential to improve the decoding of concatenated codes and indeed several papers do just this [6, 8, 11].

The main technical result today will be the following:

Lemma 5 *There exists a polynomial time algorithm to solve the weighted RS list-decoding problem, provided* $W > \sqrt{2k \sum_{i=1}^{n} \binom{w_i+1}{2}}$

At a quick glance the lemma may not seem surprising. Indeed if we set all the w_i's to 1, then the lemma is identical to Theorem 4. To see that the lemma is doing something different, notice that the lemma has a peculiar behaviour with respect to the weights w_i's — it is not *scale invariant.* On the one hand if we take all the weights, w_i's as well as W, and say double them, then the solution set of polynomials does not change. However, the lemma may not guarantee to find the solutions with the smaller weights, but might be able to guarantee to find all solutions with the larger weight. Indeed setting all weights to 2, and setting $W = \sqrt{6kn}$ gives us a list of all polynomials that agree an unweighted set of n distinct pairs in $\sqrt{\frac{3}{2}kn}$ points. Sending w_i's to infinity gives us the result we want for this section, as proven below formally.

Theorem 6 ([5]) *There exists a polynomial time to solve the* **RS list decoding problem** *in polynomial time provided* $t > \sqrt{kn}$.

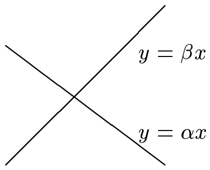
Proof Since t is an integer, we have $t \geq \lfloor\sqrt{kn}\rfloor + 1$. Set $w_1 = \cdots = w_n = w = 2nk$ (i.e., large enough that some later inequalities can be handled) and $W = t \cdot w$ and apply the algorithm from Lemma 5. If a polynomial p agrees with the point set on t points, then its weighted agreement is at least tw and Lemma 5 guarantees that this polynomial will be included in the output list provided $tw > \sqrt{nkw(w+1)}$, or in other words, provided $t > \sqrt{nk(1+1/w)} = \sqrt{nk + \frac{1}{2}}$. It can be easily verified that $\sqrt{nk + \frac{1}{2}} < \lfloor\sqrt{nk}\rfloor + 1 \leq t$ and so the conditions required for applying Lemmm 5 are indeed satisfied. ■

Thus to conclude our goal it suffices to build the algorithm that Lemma 5 claims to exist.

2.2 Algorithm

Our basic problem at this stage is to figure out how to make some algebraic sense of the weights that are given to us. I.e., how to modify the definition of the Q-polynomial so that points with larger weights constraint it more than the points with smaller weight, or to make it clear that Q "respects" points of larger weight more than the points of smaller weight. A natural idea, if we can make any formal sense of it, is to make the polynomial go through points as many times as their weight. So Q should pass through a point with weight 2 twice, while it should pass through a point of weight 10, ten times!

Let us first see an example of what it means to have a curve passing through the point $(0,0)$ twice.



The equation of this curve is $(y - \alpha x)(y - \beta x) = 0$, or $y^2 - (\alpha + \beta)xy + \alpha\beta x^2 = 0$. Intuitively, it is clear what it means for a curve to pass through the origin twice, and that the example above indeed does so. Analytically, this may be explained in terms of the partial derivatives of Q at $(0,0)$. However, since we are working over finite fields of potentially small characteristic, the partial derivatives may not be informative enough. So we'll describe the algebraic definition. To guess this definition, let us look at two familiar examples. We know what it means for a curve to pass through $(0,0)$ once - it means that the coefficient of the constant term must be zero. In the example above, where the curve passes through the origin twice, the constant term as well as the monomials of degree one (i.e., x and y) have zero as coefficients. Deducing from these cases, we guess that a polynomial Q passes through $(0,0)$ at least r times if the monomials of total degree less than r have a zero coefficient.

Now how can we extend the definition to an arbitrary point $(\alpha, \beta) \in \mathbb{F}_q \times \mathbb{F}_q$? The natural way to do this is to shift the system so that (α, β) becomes the origin. Thus we arrive at the following *definition.* (We stress that nothing is being proved so far - we are just guessing a definition; later we will need to prove that this actually works!)

Definition 7 *A polynomial $Q(x,y)$ passes through the point (α, β) at least r times if all the coefficients of monomials of total degree less than r of the polynomial $Q_{\alpha,\beta}(x,y) \stackrel{\text{def}}{=} Q(x+\alpha, y+\beta)$ are zero.*

With the above definition in hand, we now describe the optimized algorithm for weighted polynomial fitting:

RS-list-decoder-3:

Given Set of n distinct pairs $\{(x_i, y_i) | i \in [n]\}$, integer weights $w_1, \ldots, w_n$ and an integer degree bound k.

Parameter: D to be set later.

Step 1: Find a bivariate polynomial $Q(x,y)$ of $(1,k)$-weighted degree D such that:

- For every $i \in [n]$, the polynomial Q passes through the point (x_i, y_i) at least w_i times.
- $Q \not\equiv 0$.

Step 2: Factor $Q(x,y)$ and output the list of polynomials p of degree at most k provided:

- $y - p(x) \mid Q(x, y)$, and
- $p(x_i) = y_i$ for at least t values of $i \in [n]$.

To see that the new algorithm still runs in time polynomial in $\sum_{i=1}^{n} w_i$ and D, we need to show that Step 1 is solvable in so much time. The critical part of this is to show that the conditions being enforced here are still linear and homogenous. Note that the condition being enforced for Q to pass through the origin multiple times *are* linear and homogenous. What we need to establish is that the shifting of a polynomial leads to coefficients that are linear in the coefficients of Q. This is established below.

Proposition 8 *The coefficients of the polynomial $Q_{\alpha,\beta}(x, y) \stackrel{\text{def}}{=} Q(x + \alpha, y + \beta)$ are linear in the coefficients of Q.*

Proof We will prove this by giving the coefficients of $Q_{\alpha,\beta}$ explicitly. This is just an ordinary calculation. We get the coefficient of $x^i y^j$ in $Q_{\alpha,\beta}$ equals $\sum_{i' \geq i, j' \geq j} \binom{i'}{i}\binom{j'}{j} \alpha^{i'-i} \beta^{j'-j} q_{i'j'}$ where $Q(x, y) = \sum_{i,j} q_{ij} x^i y^j$. ■

Now we can go about the proof in our usual manner. We start with the observation that a non-zero polynomial does exist, provided we pick D to be large enough.

Proposition 9 *For any sequence of distinct points $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{F}_q \times \mathbb{F}_q$, and non-negative integral weights $w_1, \ldots, w_n$, there exists a non-zero polynomial Q of $(1, k)$-weighted degree at most $\sqrt{2k \sum_{i=1}^{n} \binom{w_i+1}{2}}$ such that Q passes through (x_i, y_i) at least w_i times for every $i \in [n]$.*

Proof Each condition of the form Q must pass through (x_i, y_i) at least w_i times, turns into the condition that $\binom{w_i+1}{2}$ coefficients of Q_{x_i,y_i} must be zero. In turn these turn into $\binom{w_i+1}{2}$ homogenous linear conditions on the coefficients of Q (by Proposition 8). Thus all in all we have $\sum_{i=1}^{n} \binom{w_i+1}{2}$ homogenous linear constraints. A simple calculation shows that the number of coefficients of $(1, k)$-weighted degree at most $\sqrt{2k \sum_{i=1}^{n} \binom{w_i+1}{2}}$ is more than $\sum_{i=1}^{n} \binom{w_i+1}{2}$ and thus a non-zero solution to the homogenous linear system exists. ■

So far we have proved that multiplicities can be handled. But we haven't related it to the problem at hand - why does making Q pass through a point w_i times, help weight this point w_i times?

We start to prove this below:

Lemma 10 *If $Q(x, y)$ passes through a point (α, β) at least w times and p is any polynomial such that $p(\alpha) = \beta$, then $(x - \alpha)^w$ divides the polynomial $g(x) \stackrel{\text{def}}{=} Q(x, p(x))$.*

Proof First by translation, we will turn this into a question about $(0, 0)$. Let $Q_{\alpha,\beta}(x, y) = Q(x + \alpha, y + \beta)$. Let $p_{\alpha,\beta}(x) = p(x + \alpha) - \beta$. Since $p(0) = 0$, we have x divides $p_{\alpha,\beta}(x)$. Let $g_{\alpha,\beta}(x) = Q_{\alpha,\beta}(x, p_{\alpha,\beta}(x)) = Q(x + \alpha, p(x + \alpha)) = g(x + \alpha)$. Thus $(x - \alpha)^w$ divides $g(x)$ if and only if x^w divides $g(x + \alpha) = g_{\alpha,\beta}(x)$. In other words we wish to show that the monomials of degree less than w in $g_{\alpha,\beta}$ have zero coefficient. But this is syntactically true. $Q_{\alpha,\beta}$ has no terms of total degree less than w and we are substituting for y a polynomial $p_{\alpha,\beta}(x)$ that is itself a mutliple of x. Thus all terms have degree at least w. ■

The rest of what we want is now straightforward. We will show that if for some degree k polynomial p, the weights of points (x_i, y_i) where $p(x_i) = y_i$ sum to more than the $(1, k)$-weighted degree of Q, then $y - p(x)$ must divide Q.

Lemma 11 *Let $Q(x, y)$ be any polynomial of $(1, k)$-weighted degree D that satisfies the conditions of Step 1. Suppose p is a degree k polynomial such that $\sum_{\{i \in [n] | p(x_i) = y_i\}} w_i > D$, then $y - p(x)$ divides $Q(x, y)$.*

Proof Let $g(x) = Q(x, p(x))$. Then g has degree at most D. We wish to show that g is identically zero. For a change we won't do this by showing that it has more zeroes than D but by showing that a polynomial of degree greater than D divides it (and this can happen only to the zero polynomial).

Let $S = \{i \in [n] \mid y_i = p(x_i)\}$. Note first that for $i, j \in S$, $x_i \neq x_j$ since otherwise we would have $y_i = p(x_i) = p(x_j) = y_j$ and thus $(x_i, y_i) = (x_j, y_j)$. Next note that for every $i \in S$, we have $(x - x_i)^{w_i}$ divides $g(x)$ (by Lemma 10. Putting these together, we get $\prod_{i \in S}(x - x_i)^{w_i}$ divides $g(x)$. (Here we are using the fact that the x_i's are distinct. Really what we should observe first is that the least common multiple of the $(x - x_i)^{w_i}$'s divides g. Then using the distinctness of the x_i's we should claim that the product divides g. But this would require defining least common multiple etc.) Now we are done since the polynomial $\prod_{i \in S}(x - x_i)^{w_i}$ has degree $\sum_{\{i \in [n] | p(x_i) = y_i\}} w_i > D$, thus yielding that $g \equiv 0$. ■

We may now have a proof of Lemma 5 with the algorithm described explicitly, as summarized below:

Theorem 12 ([5]) *The algorithm* **RS-list-decoder-3** *with the setting* $D = \sqrt{2k \sum_{i=1}^{n} \binom{w_i+1}{2}}$ *solves the* **Weighted RS list decoding problem** *in polynomial time provided* $W > \sqrt{2k \sum_{i=1}^{n} \binom{w_i+1}{2}}$.

Recall that as this concludes the proof of Theorem 6 which may be considered the main theorem of this lecture.

3 Towards Generalization of Decoding Algorithms

We now move towards an abstraction of the algorithm above. We motivate by considering the decoding problem for a fairly different code — the Chinese Remainder Code — lecture. We already defined this code in a previous lecture. We start by recalling the definition.

3.1 Chinese Remainder Codes and Decoding

We start with the classical "Chinese Remainder Theorem".

Theorem 13 (Chinese Remainder Theorem (CRT)) *Let $p_1, p_2, \ldots, p_k$ be positive integers that are pairwise relatively prime. Then, the map* CRT *that takes $m \in \mathbb{Z}_K$ to the k-tuple $\langle m(\bmod p_1), \ldots, m(\bmod p_k)\rangle$, where $K = \prod_{i=1}^{k} p_i$, is a bijection. Furthermore, there are polynomial time algorithms to compute* CRT *as well as its inverse.*

In other words, $m \in \mathbb{Z}_K$ is fully specified given its residues modulo $p_1, \ldots .p_k$. The CRT code is based on the observation that specifying $m \in \mathbb{Z}_K$ modulo a large number of relatively prime integers (larger than the minimum required) gives a redundant representation of m. We formalize this next.

Let $p_1 < p_2 < \ldots < p_n$ be primes, and let $K = \prod_{i=1}^{k} p_i$ and $N = \prod_{i=1}^{n} p_i$ where $k < n$. Then, $\langle m(\bmod p_1), \ldots, m(\bmod p_n)\rangle$ is a redundant *encoding* of m. In particular, given any k out of the n possible residues, we can recover m uniquely. It follows that the encoding of any two message must differ in at least $k - 1$ positions, so this encoding yields a "$(n, k, n - k + 1)$ code". This code motivates the natural (list-)decoding problem.

PROBLEM Given $p_1 < \ldots < p_n$, $k < n$ and a codeword corresponding to the residues $\langle r_1, \ldots, r_n\rangle$.
TASK Find a $m \in \{0, 1, \ldots, \prod_{i=1}^{k} p_i - 1\}$ such that $m(\bmod p_i) = r_i$ for many values of i.

> **CRT List-Decoding Problem:**
> GIVEN: $p_1 < \ldots < p_n$, integer K, a vector of residues $\langle r_1, \ldots, r_n\rangle$, and an agreement parameter t.
> GOAL Find a list of all $m \in \mathbb{Z}_K$ such that, $m(\bmod p_i) = r_i$ for at least t values of $i \in [n]$.

Our plan to get a decoding algorithm for this problem is by the following sequence of steps: First, we will find a common generalization of the Reed-Solomon and CRT codes; Next, we will "lift" our algorithm for decoding for RS decoding to this common generalization. This won't quite give us an algorithm for the common generalization, but an outline of one. Finally, we will specialize the "algorithm" to the case of CRT codes and then prove its efficiency and correctness. Today we will move towards the first of these steps.

3.2 Algebra Review

The Chinese Remainder Codes and the Reed-Solomon Codes may be viewed as examples of a more general class of error-correcting codes based "ideals" in commutative rings. We start by recalling some of the jargon.

- A commutative ring R is an *integral domain* if it does not contain any zero divisors, i.e., there do not exist non-zero elements $a, b \in R$ such that $a \cdot b = 0$.
- An *ideal* I in a ring R is a subset of R closed under addition and general multiplication (by any element in R), i.e., for any $a, b \in I$ and $r \in R$, it is the case that $a + b, ar \in I$.

Ideals in commutative rings are nice objects to work with. First, they capture the "modulo" operation (as formalized in the next definition). The collection of ideals in a ring is interesting to study since they are closed under some natural (and some slightly concocted) operations.

- Let I be an ideal of a ring R. The cosets of the additive subgroup of I^+ of R^+ are the subsets $a + I, a \in R$. The set of cosets R/I forms a group with a ring structure inherited from R. This ring is known as the *quotient ring* of R over the ideal I.

The quotient ring R/I of a ring R over an ideal I forms the basis of a modular reduction. This is the reduction that maps an element $a \in R$ to the coset $a+I$. However, we (especially the computer scientists among us) are used to thinking about modular reductions in slightly different language. First, given an ideal I, we usually pick a set $\Sigma \subseteq R$ of representatives for the quotient ring R/I, i.e., for every $a \in R$, there exists exactly one $b \in \Sigma$ such that $b \in a + I$. Note that the one-to-one correspondence between R/I and Σ gives a ring structure to Σ. Having picked such a set, we use a slightly different notation $a(\mathrm{mod} I)$, or just $a(I)$, to denote the element $b \in \Sigma$ such that $b \in a + I$. If I is the ideal generated by a single element $p \in R$ (i.e., $I = \{pr \mid r \in R\}$), then this gives the familiar $(\mathrm{mod} p)$ operation.

The choice of repreesentatives does not alter the algebra underneath. So we will often leave this choice unspecified. However representations are always important to *computation*. Unless we fix a set Σ and representations of elements in Σ, computational aspects can *not* be clarified. We now go on to other properties of ideals.

Proposition 14 *Let I and J be ideals of a ring R. Then their* sum, *denoted $I + J$,* intersection, *denoted $I \cap J$, and product, denoted IJ and defined to be $IJ = \{\sum_i a_i b_i \mid a_i \in I, b_i \in J\}$, are also ideals of R.*

3.3 Ideal Error-Correcting Codes

We now describe how the language of ideals can be used to define error-correcting codes.

Definition 15 *An* ideal error-correcting code *C is given by an integral domain R, a sequence of n ideals $I_1, \ldots, I_n \subseteq R$ (and associated sets of representatives $\Sigma_1, \ldots, \Sigma_n$), and a* message space *$M \subseteq R$. The canonical encoding function for C maps the message $m \in M$ to the sequence $\langle m(I_1), \ldots, m(I_n)\rangle$.*

Some of the previous choice of notation may become clearer now: Usually $\Sigma_1 = \cdots = \Sigma_n = \Sigma$ and then the code maps M to Σ^n. But is other cases Σ_i's may be different (as in the case of the CRT code). For practical usage Σ_i's better be finite. However the mathematics does not enforce such a restriction. We now see how Reed-Solomon codes and CRT codes are both special cases of ideal error-correcting codes.

1. Chinese Remainder Codes. This corresponds to $R = \mathbb{Z}$, $I_1 = p_1\mathbb{Z}, \ldots, I_n = p_n\mathbb{Z}$, and $M = \{0, 1, \ldots, \prod_{i=1}^{k} p_i - 1\}$. The representative set Σ_i representing $\mathbb{Z}/p_i\mathbb{Z}$ is the natural one, i.e., $\Sigma_i = \mathbb{Z}_{p_i} = \{0, \ldots, p_i - 1\}$.
2. Reed-Solomon Codes. This corresponds to $R = \mathbb{F}_q[x]$, $I_1 = (x - \alpha_1), \ldots, I_n = (x - \alpha_n)$ where $\alpha_1, \ldots, \alpha_n$ are the n distinct elements of $\mathbb{F}_q$ that we pick to evaluate the polynomial. M is the set of polynomials in R of degree less than k. The choice of representatives is again the natural one with $\Sigma_1 = \cdots = \Sigma_n = \mathbb{F}_q$, where $\mathbb{F}_q$ stands for the set of degree zero polynomials in R.

In the next lecture we will see how to lift our decoding algorithm(s) and then specialize them to get a decoding algorithm for the CRT code.

Bibliographic notes

The optimized version of the Reed-Solomon decoding algorithm that worked if the agreement parameter $t > \sqrt{2kn}$ is from [13]. A careful analysis of this algorithm for every choice of rate is given in [14]. The improved algorithm working if $t > \sqrt{kn}$ is due to Guruswami and Sudan [5].

Chinese remainder codes have been studied for a long time now. Early papers on this code include those of Asmuth and Bloom [1] and Watson [15]. Even some books have been written on this topic (see Krishna et al. [9] and Soderstrand et al. [12]). References for decoding algorithms for this code include [10, 3, 2, 4]. We will discuss their contributions in further detail in the next lecture. For now, we mention that the formalization of "ideal error-correcting codes" is from [4].

References

[1] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29:208–210, March 1983.

[2] Dan Boneh. Finding smooth integers in short intervals using CRT decoding. *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 265–272, 2000.

[3] Oded Goldreich, Dana Ron, and Madhu Sudan. Chinese remaindering with errors. *IEEE Transactions on Information Theory*, 46(5):1330–1338, July 2000. Extended version appears as ECCC Technical Report TR98-062 (Revision 4), `http://www.eccc.uni-trier.de/eccc`.

[4] Venkatesan Guruswami, Amit Sahai, and Madhu Sudan. Soft-decision decoding of Chinese Remainder codes. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, pages 159–168, Redondo Beach, California, 12-14 November 2000.

[5] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory*, 45:1757–1767, 1999.

[6] Venkatesan Guruswami and Madhu Sudan. List decoding algorithms for certain concatenated codes. *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 181–190, 2000.

[7] G. David Forney Jr. Generalized minimum distance decoding. *IEEE Transactions on Information Theory*, 12(2):125–131, April 1966.

[8] Ralf Kötter and Alexander Vardy. Algebraic soft-decision decoding of Reed-Solomon codes. *Proceedings of the 38th Annual Allerton Conference on Communication, Control and Computing*, pages 625–635, October 2000.

[9] H. Krishna, B. Krishna, K.-Y. Lin, and J.-D. Sun. *Computational Number Theory and Digital Signal Processing: Fast Algorithms and Error Control Techniques.* CRC Press Inc., Boca Raton, Florida, 1994.

[10] David M. Mandelbaum. On a class of arithmetic codes and a decoding algorithm. *IEEE Transactions on Information Theory*, 21(1):85–88, January 1976.

[11] Rasmus R. Nielsen. Decoding concatenated codes using Sudan's algorithm. *Manuscript submitted for publication*, May 2000.

[12] M.A. Soderstrand, W.K. Jenkins, G.A. Jullien, and F.J. Taylor. *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing.* IEEE Press, New York, 1986.

[13] Madhu Sudan. Decoding of Reed-Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

[14] Madhu Sudan. Decoding of Reed-Solomon codes beyond the error-correction diameter. *Proceedings of the 35th Annual Allerton Conference on Communication, Control and Computing*, 1997.

[15] R. W. Watson and C. W. Hastings. Self-checked computation using residue arithmetic. In *Proceedings of the IEEE*, volume 44, pages 1920–1931, December 1966.