Deterministic List Decoding of Reed Solomon Codes

Soham Chatterjee

May 15, 2026

Tata Institute of Fundamental Research

Joint work with

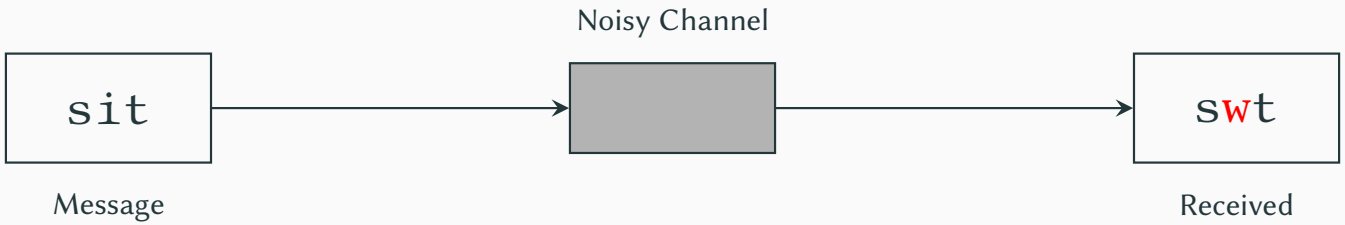
To appear in STOC 26

Introduction

What is Coding Theory?

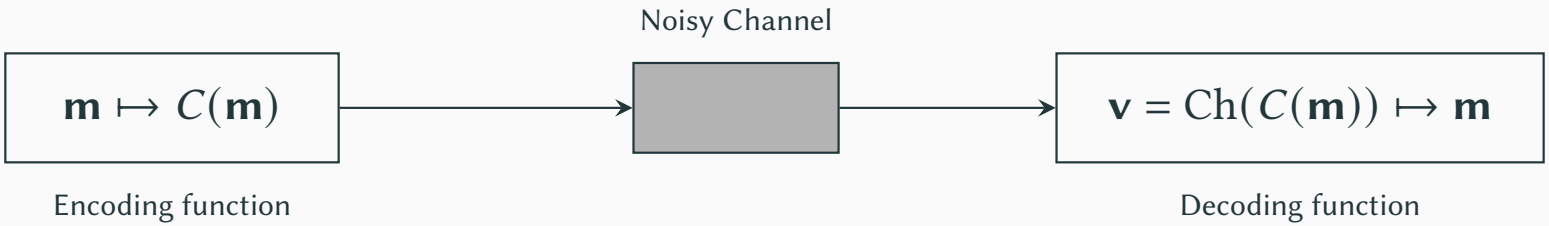
Suppose you want to send a message to your friend.

But the channel is **noisy**. Some symbols can change arbitrarily.



The symbol `i` got corrupted to `w`. What was the original message? `sit` or `sat`.

Solution: Encode m before sending; **decode v** after receiving.



A Simple Idea: Repetition Code

Idea: Send the same message 3 times: `sit` $\longrightarrow$ `sitsitsit`.

Decoding by majority

For each coordinate, take the **majority** over the three copies. A single symbol-flip anywhere is corrected.

Problem: The codeword is $3\times$ as long as the message. To handle more errors we'd need even more copies. This is **too wasteful**!

Solution

We want a **smarter encoding** that corrects errors without a large blowup in length.

With this in place, let us set up some notation…

Key Assumption

The number of errors is **bounded** — the channel cannot corrupt the entire word.

Introduction to Coding Theory

We now introduce a few terminologies used in coding theory.

Let $\mathcal{M}$ be the **message space** and Σ a finite **alphabet**. A code is a family of maps, one for each length of codeword:

- **Encoding:** $\mathrm{Enc}_n : [|\mathcal{M}|] \to \Sigma^n$
- **Decoding:** $\mathrm{Dec}_n : \Sigma^n \to [|\mathcal{M}|]$

The image $C = \mathrm{Enc}_n([|\mathcal{M}|]) \subseteq \Sigma^n$ is the **code**; its elements are **codewords**.

- **Blocklength:** n
- **Dimension:** $k = \log |C|$
- **Rate:** $R(C) = \dfrac{k}{n \log |\Sigma|}$

Rate is something like how much information of the message you are sending per bit.

Introduction to Coding Theory

The **Hamming distance** between $u, v \in \Sigma^n$ is the number of coordinates where they differ: $\Delta(u, v) = \left|\{ i : u_i \neq v_i \}\right|$

coordinate	1	2	3	4	5	6
u	0	1	1	0	1	0
v	1	1	0	0	1	1
	$\neq$	$=$	$\neq$	$=$	$=$	$\neq$

$\Rightarrow \Delta(u, v) = 3$

- **Distance of code:** $\Delta(C) = \min_{c_1 \neq c_2 \in C} \Delta(c_1, c_2)$
- **Relative distance:** $\delta(C) = \dfrac{\Delta(C)}{n}$

Intuitively: a large distance means codewords are far apart, making it easier to recover from errors.

The set $\{u \in \Sigma^n : \Delta(u, v) \leq r\}$ is a **Hamming ball** of **radius** r around v. We often speak of decoding *radius* rather than distance.

Introduction to Coding Theory

Goal: Construct codes such that

- Codewords to be "far apart" from each other $\implies$ **High Distance**.
- Redundancy to be low $\implies$ **High Rate**.

Relation between Rate and Distance (Singleton Bound)

For any code C

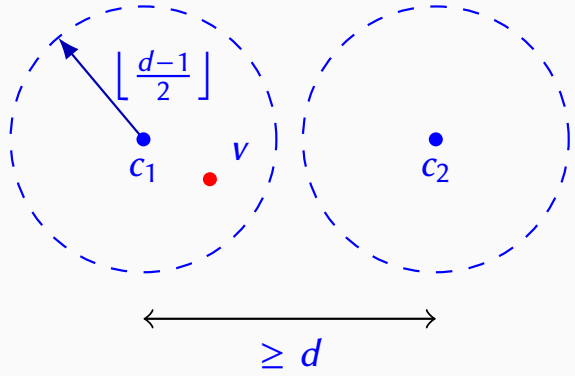
$$k \leq n - d + 1$$

Asymptotically $R + \delta \leq 1$ as n becomes very large.

Codes achieving this bound are called **Maximum Distance Separable (MDS)** codes.

Unique Decoding

Let $\Delta(C) = d$. For any $v \in \Sigma^n$ there is at most one codeword $c \in C$ such that $\Delta(v, c) \leq (d-1)/2$.



Unique Decoding Problem: Given a received word $v \in \Sigma^n$, find the unique codeword $c \in C$ such that $\Delta(v, c) < d/2$ if it exists.

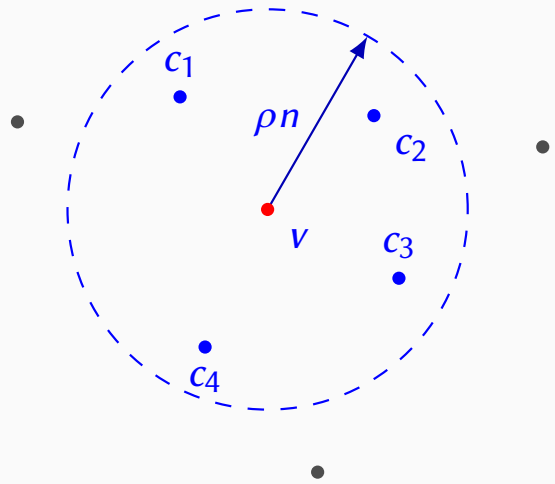
- If we go more than $d/2$ distance, multiple codewords may lie in the radius of hamming ball.

List Decoding

Definition ((ρ, L)-List Decodable)

C is (ρ, L)-list decodable if for every $v \in \Sigma^n$,

$$|\{c \in C \mid \Delta(c, v) \leq \rho n\}| \leq L$$



We write $L(v, \rho) = \{c \in C \mid \Delta(c, v) \leq \rho n\}$.

Unlike unique decoding, **multiple codewords** may lie inside the ball.

We sometimes say "m is in the list" to mean $\mathrm{Enc}_n(f) \in L(v, \rho)$, depending on the context.

List Decoding

List Decoding Problem: Given a received word $v \in \Sigma^n$, compute $L(v, \rho)$.

- **Combinatorial:** Prove $|L(v, \rho)| = \text{poly}(n)$.
- **Algorithmic:** Find all codewords in $L(v, \rho)$ in $\text{poly}(n)$ time.

Theorem (Johnson Bound)

For a code C with rate R, the list size is polynomial in n for $\rho \leq 1 - \sqrt{R}$ (also called Johnson radius).

Convention: We use **agreement** $t = 1 - \rho$ in place of error fraction ρ.

Why Polynomial-Based Codes?

We want codes with **high rate** and **large distance**. A natural question: what mathematical structure should we build on?

Answer: Polynomials are nice to use.

- The algebra of polynomials is classical and rich.
- Polynomial arithmetic can all be done very efficiently. All the basic operations in near linear time.
- Connections to algebra, number theory, and algebraic geometry give powerful tools for both encoding and decoding.

Examples: Reed-Solomon Codes, Reed-Müller Codes, BCH Codes, Multiplicity Codes…

Reed Solomon Codes (RS)

Construction

Fix a finite field $\mathbb{F}_q$ and distinct evaluation points $S = \{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbb{F}_q$.

$RS[n, k]_q$: each message m is identified with a polynomial $f_m \in \mathbb{F}_q[X]$ of degree $< k$,

and encoded by evaluating at all points of S:

$$\mathrm{Enc}(m) = \big(f_m(\alpha_1), f_m(\alpha_2), \ldots, f_m(\alpha_n)\big)$$

Rate: k/n.

Codewords form a **vector space**, so $\Delta(C)$ = min nonzero positions of any nonzero codeword = min nonzero evaluations of a nonzero $\deg < k$ polynomial.

Distance: $d = n - k + 1$

This meets the Singleton bound $d \leq n - k + 1$, making RS codes **MDS**.

Reed Solomon Decoding History

Goal: Decode in $\mathrm{poly}(n, \log q)$ time or $\mathrm{poly}(n)$ field operations.

Unique Decoding Radius: $> (n + k - 1)/2$

Johnson Radius: agreement $> \sqrt{n(k-1)}$.

Good news: all three settings above admit $\mathrm{poly}(n)$ algorithms.

Algorithm	**Agreement**	**Type**
Berlekamp–Welch	$> (n + k - 1)/2$	deterministic
Sudan '97	$> \sqrt{2n(k-1)}$	randomized
Guruswami–Sudan '99	$> \sqrt{n(k-1)}$	randomized

The **deterministic** variant of GS runs in $\mathrm{poly}(n, |\mathbb{F}|)$.

Polynomial dependence on the field characteristic ‼.

Framework of Sudan and Guruswami-Sudan

Let $w = \{(\alpha_j, \beta_j)\}_{j \in [n]}$ be the received word. Both algorithms share the same two-step structure:

Step 1: Interpolation. Find a nonzero bivariate polynomial $Q(X, Y) \in \mathbb{F}_q[X, Y]$ of $(1, k-1)$ degree D vanishing at every received point (α_j, β_j) with some multiplicity, chosen.

So that every close codeword f satisfies $Y - f(X) \mid Q(X, Y)$.

(a, b) degree of $X^i Y^j$ is $ai + bj$. So take max for polynomials.

Sudan uses multiplicity 1*; Guruswami–Sudan uses higher multiplicity* $m > 1$ *to push the radius further.*

Step 2: Factorization. Completely factor $Q(X, Y)$ over $\mathbb{F}_q$ and collect all factors of the form $Y - f(X)$

Factorization Barrier

The **interpolation step is already deterministic**. The bottleneck is the **factorization step**.

Both Sudan and Guruswami–Sudan algorithms rely heavily on the **factorization of bivariate polynomials** over finite fields.

In fact, the difficulty already appears in the **univariate** case: factoring a degree-d polynomial over $\mathbb{F}_q$ deterministically in $\mathrm{poly}(d, \log q)$ time is **open**.

- **Berlekamp**, **Cantor–Zassenhaus**: randomized $\mathrm{poly}(d, \log q)$ algorithms for univariate factoring over $\mathbb{F}_q$.
- Deterministic variants run in time **polynomial in the field characteristic**. Too slow when the characteristic is large (super polynomial)

State of Art: No truly deterministic $\mathrm{poly}(n, \log q)$ factorization or RS-list decoding algorithm is known.

Our Result

We give the **first truly deterministic** polynomial-time list decoding algorithm for Reed–Solomon codes, matching the running time of the Sudan and Guruswami–Sudan algorithms.

Key contribution: A deterministic algorithm for the factorization step: finding all factors of $Q(X, Y)$ of the form $Y - f(X)$ in $\mathsf{poly}(n)$ field operations, with no dependence on the field characteristic.

Theorem ([CHK25])

There is a deterministic algorithm that, for every finite field $\mathbb{F}$ and parameters $n > k \in \mathbb{N}$, runs in time $\mathsf{poly}(n, \log |\mathbb{F}|)$ and list decodes $RS[n, k]_{\mathbb{F}}$ from agreement greater than $\sqrt{(k-1)n}$.

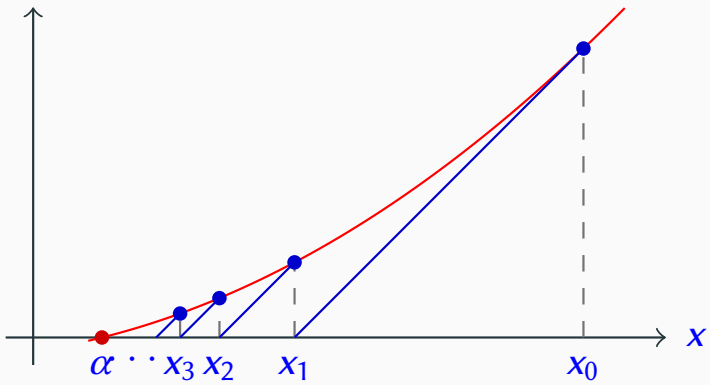
Derandomization of Sudan

Newton Iteration

Starting from x_0, each step draws a line with slope $f'(0)$ at $(x_t, f(x_t))$ and sets x_{t+1} to its x-intercept.

Each step:

1. $f'(0)$ should be non-zero
2. Evaluate $f(x_t)$ and $f'(0)$
3. $x_{t+1} = x_t - \dfrac{f(x_t)}{f'(0)}$



Converges linearly to the root α.

The same idea works for bivariate polynomials over $\mathbb{F}_q$, replacing reals with formal power series. But we take modulo an ideal.

Sudan Derandomization

Let $w = (\alpha_i, \beta_i)_{i \in [n]}$ is received word. We interpolate a polynomial $Q(X, Y)$ of $(1, k-1)$ degree $\sqrt{2n(k-1)}$ vanishes at every (α_i, β_i).

Let f is in the list. And let $j \in [n]$ such that $f(\alpha_j) = \beta_j$. Suppose Q is the polynomial from interpolation step.

If $\partial_Y Q(\alpha_j, f(\alpha_j)) \neq 0$: Then Newton Iteration from $Y_0 = f(\alpha_j)$ till Y_k gives f.

Else $\partial_Y Q(\alpha_j, f(\alpha_j)) = 0$ for all j in agreement.

Observe: $\partial_Y(Q(X, f(X)))$ has more than $\sqrt{2n(k-1)}$ roots but degree at most $\sqrt{2n(k-1)}$. Implying $\partial_Y(Q(X, f(X))) \equiv 0$

So recurse on $\partial_Y(Q(X, f(X)))$.

✓ Done.

Derandomization of Guruswami-Sudan

Local Splitting

Let $P(X, Y) \in \mathbb{F}_q[X, Y]$ with no-pure X-factors. Let $(\alpha, \beta) \in \mathbb{F}_q^2$ any point.

Suppose we are given the factorization $P = \prod_{i=1}^{s} P_i$ into irreducibles (with multiplicity)

Eg: $P(X, Y) = (Y^2 + X)(Y^2 + X + 1)^2$ then

$$P_1 = Y^2 + X, \quad P_2 = Y^2 + X + 1, \quad P_3 = Y^2 + X + 1$$

We partition $[s]$ into four sets which defines four types of factors at (α, β):

$$A(\alpha, \beta), \quad B(\alpha, \beta), \quad C(\alpha, \beta), \quad D(\alpha, \beta)$$

Let's call them A, B, C, D.

Local Splitting

Let $P = (Y^2+X+1)(Y^2+X)(Y^2+X^2+Y)(XY+1)$, $(\alpha, \beta) = (0, 0)$.

Set	Condition on $P_i(\alpha, Y)$	Example
A	$P_i(\alpha, \beta) \neq 0,\ \deg_Y P_i(\alpha, Y) \geq 1$	$Y^2 + X + 1$
B	$P_i(\alpha, Y) = \gamma(Y - \beta)^m,\ m \geq 1,\ \gamma \neq 0$	$Y^2 + X$
C	$P_i(\alpha, Y) = (Y-\beta)^m \hat{P}_i(Y),\ \hat{P}_i(\beta) \neq 0,\ \deg \hat{P}_i \geq 1$	$Y^2 + X^2 + Y$
D	$P_i(\alpha, \beta) \neq 0,\ \deg_Y P_i(\alpha, Y) = 0$	$XY + 1$

Define $P_A = \prod_{i \in A} P_i$ and similarly P_B, P_C, P_D.

Observe: If P is monic in Y then D is empty.

A Nice Observation

We interpolate a polynomial $Q(X, Y)$ of $(1, k-1)$ degree $m\sqrt{2n(k-1)}$ vanishes at every (α_i, β_i) with multiplicity $m \approx \sqrt{n(k-1)}$.

For any $j \in [n]$:

If $f(\alpha_j) = \beta_j$: Then $Y - f(X) \mid P_B$.

If $f(\alpha_j) \neq \beta_j$: Then $Y - f(X) \mid P_A$.

Remark

We will assume Q is monic in Y and has no pure X-factors.

Derandomization

If only we had: An efficient $\mathsf{poly}(n, \log q)$ time algorithm $\textsc{Split}$ to find P_A, P_B from $P, (\alpha, \beta)$ then:

Key insight: We need not fully factor Q.

If we knew **which points** have $f(\alpha_j) = \beta_j$, we can directly **interpolate** f. No factorization needed ‼

Algorithm:

1. $S \longleftarrow \{Q\}$
2. Choose $j \in [n]$ and for all $g \in S$ compute $(g_A, g_B) = \textsc{Split}(g, (\alpha_j, \beta_j))$
3. Remove g from S and put g_A, g_B in S.
4. Continue from step 2 till S stabilizes.
5. Do some interpolations to recover list.

Recover List from Stable Set

Observe: If f is in the list there is one factor $g \in S$, $Y - f(X) \mid g$.

Lemma

For all $j \in [n]$,

$$g(\alpha_j, \beta_j) = 0 \iff f(\alpha_j) = \beta_j$$

So go over all $g \in S$, find $j \in [n]$ such that $g(\alpha_j, \beta_j) = 0$, do interpolation to find appropriate f.

Problem: Stabilization can take **exponentially long**.

Notice: Every Split produces two factors, and any two elements of S share no common factor at any stage.

$\implies$ Each step **partitions** the irreducible factors of Q.

✓ Done.

Splitting Algorithm

Hensel Lifting

Given $P \equiv gh \bmod (X-\alpha)^m$ with g, h coprime mod $(X-\alpha)$, lift the factorization to twice the precision.

Theorem (Hensel)

There exist unique g', h' with $g' \equiv g$, $h' \equiv h \bmod (X-\alpha)^m$ and

$$P \equiv g'h' \bmod (X-\alpha)^{2m}$$

computable in $\mathsf{poly}(\deg P, m)$ field operations.

Iterating $\lceil \log m \rceil$ times gives a full factorization mod $(X-\alpha)^m$.

Non-monic version: [Sinhababu–Thierauf, 2021]. We extend with degree bounds for iterated lifting.

Algorithm: Lifting

$P(X, Y) \in \mathbb{F}_q[X, Y]$, monic in Y with no pure X-factors and point (α, β).

We factorize:

$$P(\alpha, Y) = \underbrace{(Y - \beta)^m}_{g_0} \cdot \underbrace{\hat{P}(Y)}_{h_0}, \quad \hat{P}(\beta) \neq 0$$

If $(Y - \beta)^m = P(\alpha, Y)$ then $P = P_A$

If $\hat{P}(Y) = P(\alpha, Y)$ then $P = P_B$

Otherwise:

Use Hensel Lifting $t = 2\log(\deg_Y P)$ times to get g_t, h_t such that

$$P \equiv g_t h_t \bmod (X - \alpha)^{2^t}, \quad g_t \equiv g_0 \bmod (X - \alpha), \quad h_t \equiv h_0 \bmod (X - \alpha)$$

Algorithm: Recursive Step

g_t, h_t may not be actual factors of P as we are viewing modulo $(X-\alpha)^{2^t}$.

Observe: Actual factor, g', h' such that $g_t = g' \cdot h''$, $h' = h'' \cdot h_t$.

E.g. $P = (Y^2 + X - 1)(Y - 1)$ at $\alpha = 0$.

Mod X: $P \equiv (Y-1)^2(Y+1)$, so Hensel gives $g_t = (Y-1)^2$, $h_t = (Y+1)$.

The extra $(Y-1)$ in g_t comes from irreducible $Y^2 + X - 1$, so $h'' = (Y-1)$.

Need to solve linear systems of the form:

$$F \equiv E \cdot h_t \bmod (X-\alpha)^{2^t}, \qquad V \equiv U \cdot g_t \bmod (X-\alpha)^{2^t}$$

Take $\deg_Y(F, V) \leq \deg_Y(P) - 1$, $\deg_X(F, V) \leq \deg_X(P)$ to ensure factors.

Observe: Both F, V have factors of P but not exactly factor of P.

So we take gcd $P_1 = gcd(P, F)$, $P_2 = gcd(P, V)$

Recurse on $P_1, P/P_1$. Then combine them to get P_A, P_B.

Final Theorem

Lemma

If the algorithm passes initial checks and has no solution of linear systems. Then $P = P_C$.

Here we mention the full version of the theorem we proved:

Theorem

For every bivariate polynomial $P(X, Y) \in \mathbb{F}_q[X, Y]$ *and point* $(\alpha, \beta) \in \mathbb{F}_q^2$ *the above algorithm outputs* (P_1, P_2) *such that* $P_1 = P_A \cdot R_1$ *and* $P_2 = P_B \cdot R_2$ *where* $R_1 R_2 \mid P_D$.

Open Problems

- Derandomize Sudan and Guruswami-Sudan in near linear time.
- What about beyond johnson bound list decoding of Reed Solomon Codes?
  Ben-Sasson-Kopparty-Radhakrishnan (2006) showed us we have no hope for low rate regime. But high/constant rate we have hope
- For beyond Johnson Bound even finding one element of the list is open.
- List decode Univariate Multiplicity Codes deterministically upto list decoding capacity in $\mathsf{poly}(1/\epsilon)\tilde{O}(n)$ time.

Thank You