Demystifying the Mathematics of Erasure Coding

FOSDEM 2026

Gerlind Deschner

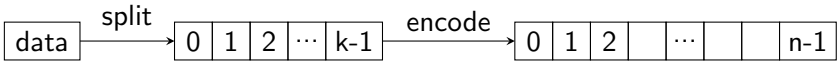
January 31, 2026

About me

- Gerlind Deschner
- Bachelor's degree in Mathematics from Technical University Berlin
- Currently finishing Master's degree in Mathematics at University of Oslo
- Big fan of FOSDEM

What is Erasure coding

- A redundancy technique for storage systems $\rightarrow$ data protection
- Enabling storage efficiency, at the expense of incurring a computational overhead
- Split data in k chunks, use encoding to get $n \geq k$ chunks ($n - k$ redundant chunks) $\rightarrow$ store across different locations



- Importantly: considers transient and permanent failures, not errors
- Less redundancy than replication, but same reliability
- Application: production storage systems, e.g. cloud storage

History

- 1960 paper "polynomial codes over certain finite fields" by Reed and Solomon publish (telecommunication)
- 1980s Redundant Array of Independent/Inexpensive Disks (RAID)
- 2000s applied distributed storage systems and cloud computing

Reed-Solomon-Code Fundamentals

Consider storing 'helloworld'

Reed-Solomon-Code Fundamentals

Consider storing 'helloworld'

1. Split into $k = 5$ chunks (same size)

'he'	'll'	'ow'	'or'	'ld'

Reed-Solomon-Code Fundamentals

Consider storing 'helloworld'

1. Split into $k = 5$ chunks (same size)

'he'	'll'	'ow'	'or'	'ld'

2. Encode with $p(x)$ into $n = 8$ chunks (same size)

$p(0)$	$p(1)$	$p(2)$	$p(3)$	$p(4)$	$p(5)$	$p(5)$	$p(6)$	$p(7)$

$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{8}$

Reed-Solomon-Code Fundamentals

Consider storing 'helloworld'

1. Split into $k = 5$ chunks (same size)

'he'	'll'	'ow'	'or'	'ld'

2. Encode with $p(x)$ into $n = 8$ chunks (same size)

$p(0)$	$p(1)$	$p(2)$	$p(3)$	$p(4)$	$p(5)$	$p(5)$	$p(6)$	$p(7)$

$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{8}$

3. Can recover 'helloworld' from any 5 of the 8 chunks!

Reed-Solomon-Code Fundamentals

Consider storing 'helloworld'

1. Split into $k = 5$ chunks (same size)

'he'	'll'	'ow'	'or'	'ld'

2. Encode with $p(x)$ into $n = 8$ chunks (same size)

$p(0)$	$p(1)$	$p(2)$	$p(3)$	$p(4)$	$p(5)$	$p(5)$	$p(6)$	$p(7)$

$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{8}$

3. Can recover 'helloworld' from any 5 of the 8 chunks!

But what is $p(x)$? How do we recover?

Polynomials

A polynomial $p(x)$ is a function

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}$$

with real valued coefficients $a_0, a_1, \ldots, a_{k-1}$ for a positive integer k.

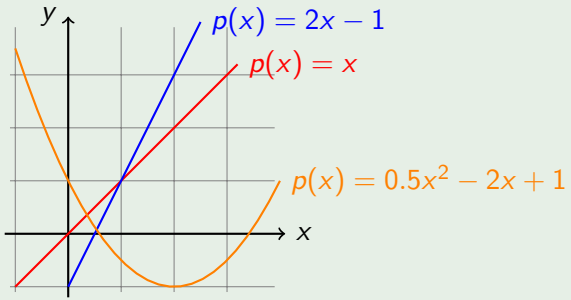
Polynomials

A polynomial $p(x)$ is a function

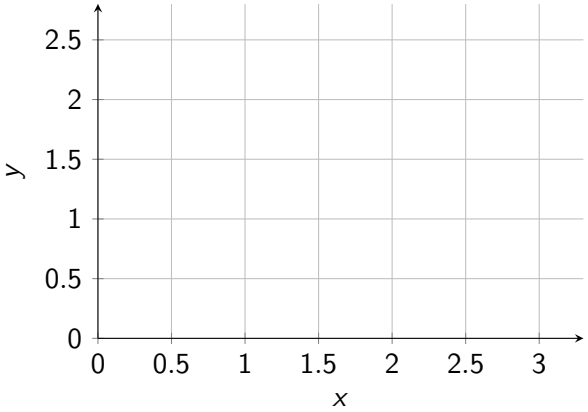
$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}$$

with real valued coefficients $a_0, a_1, \ldots, a_{k-1}$ for a positive integer k.

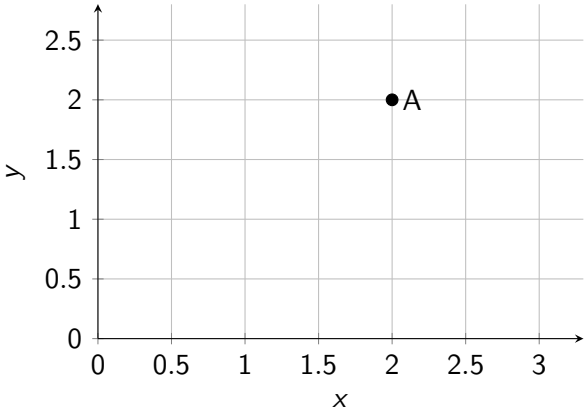
Example



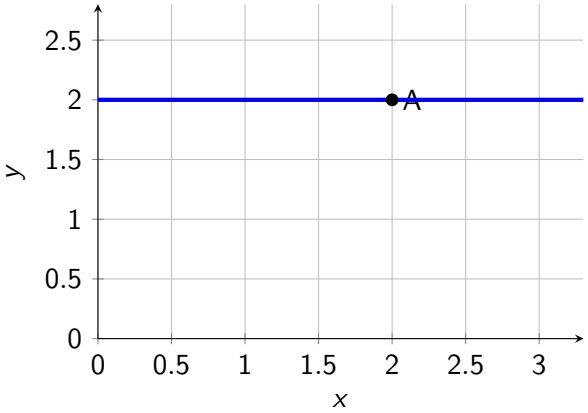
Now approach a fundamental mathematical result by drawing polynomials through points



1. Look at one point and polynomials $p(x) = a_0$

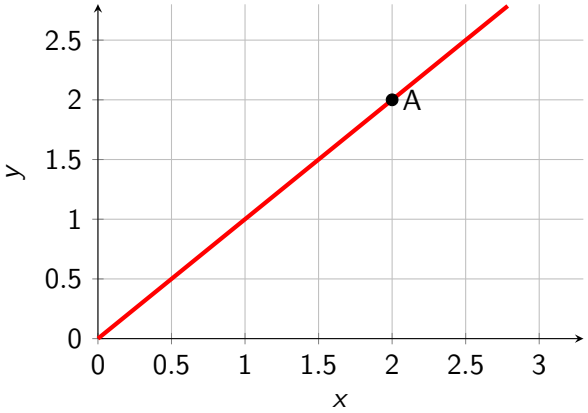


1. Look at one point and polynomials $p(x) = a_0$



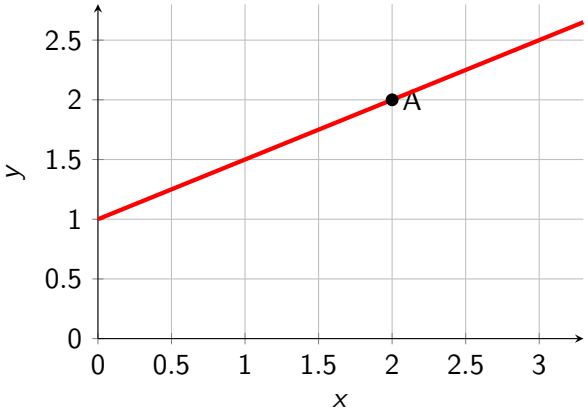
$p(x) = 2$ unique going through A!

2. Look at one point and polynomials $p(x) = a_0 + a_1 x$



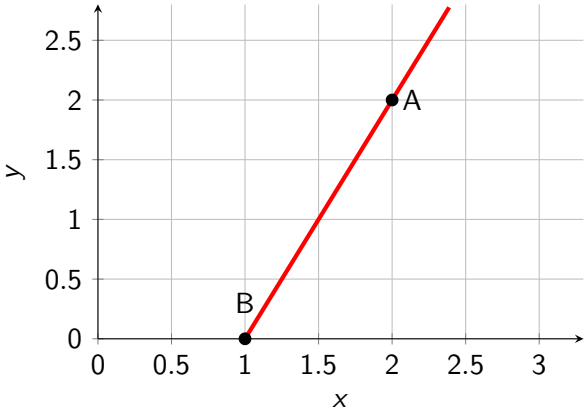
$p(x) = x$

2. Look at one point and polynomials $p(x) = a_0 + a_1 x$



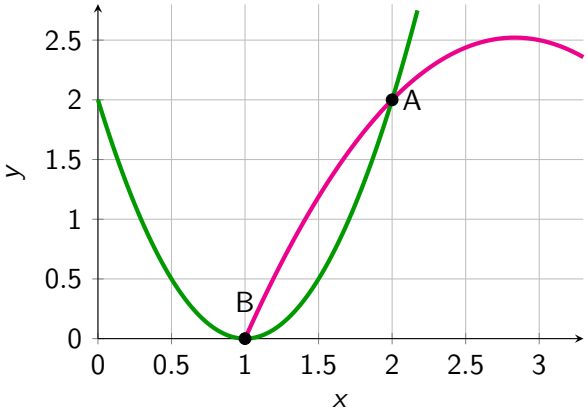
$p(x) = 1 + 0.5x$

3. Look at two points and polynomials $p(x) = a_0 + a_1 x$



$p(x) = -2 + 2x$ unique going through A and B

4. Look at two points and polynomials $p(x) = a_0 + a_1 x + a_2 x^2$



$p(x) = 2 - 4x + 2x^2$
$p(x) = -3.5 + 4.25x - 0.75x^2$

Observation

Observed that we need

- one point (x_0, y_0) to uniquely determine $p(x) = a_0$ with $p(x_0) = y_0$,
- two points (x_0, y_0) and (x_1, y_1) to uniquely determine $p(x) = a_0 + a_1 x$ with $p(x_0) = y_0$ and $p(x_1) = y_1$,
- three points (x_0, y_0) and (x_1, y_1) and (x_2, y_2) to uniquely determine $p(x) = a_0 + a_1 x + a_2 x^2$ with $p(x_0) = y_0$ and $p(x_1) = y_1$ and $p(x_2) = y_2$,
- ....

Theorem

Given coordinates $(x_0, y_0), \ldots, (x_{k-1}, y_{k-1})$ *with distinct x-coordinates, there exists a unique polynomial* $p(x) = a_0 + \cdots + a_{k-1}x^{k-1}$ *with* $p(x_0) = y_0, \ldots, p(x_{k-1}) = y_{k-1}$.

Equivalently:

Theorem

Given k distinct points $x_0, \ldots, x_{k-1}$, *each polynomial* $p(x) = a_0 + \cdots + a_{k-1}x^{k-1}$ *is uniquely determined by its values* $p(x_0) = y_0, \ldots, p(x_{k-1}) = y_{k-1}$.

But how does this help us $\rightarrow$ encoding $p(x)$ is a polynomial defined through the k chunks of our data. (details now)

Encoding

- Data in $k = 5$ chunks

'he'	'll'	'ow'	'or'	'ld'

- Define the encoding $p(x)$ as

$$p(x) = m_0 + m_1 x + \cdots + m_{k-1} x^{k-1}$$

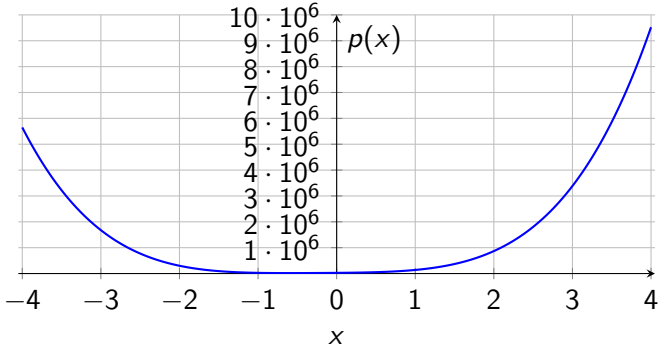
where m_i is the number represented by the whole of chunk i. Meaning for our example

- $m_0 = 26725$ the decimal number of 'he'
- $m_1 = 27756$ the decimal number of 'll'
- $m_2 = 28535$ the decimal number of 'ow'
- ...

giving

$$p(x) = 26725 + 27756x + 28535x^2 + 28530x^3 + 27748x^4$$

$$p(x) = 26725 + 27756x + 28535x^2 + 28530x^3 + 27748x^4$$



What do we do with the polynomial? And why did we need the mathematical result?

Theorem

Given k distinct points $x_0, \ldots, x_{k-1}$, each polynomial $p(x) = a_0 + \cdots + a_{k-1}x^{k-1}$ is uniquely determined by its values $p(x_0) = y_0, \ldots, p(x_{k-1}) = y_{k-1}$.

We get that

$$p(x) = m_0 + \cdots + m_{k-1}x^{k-1}$$

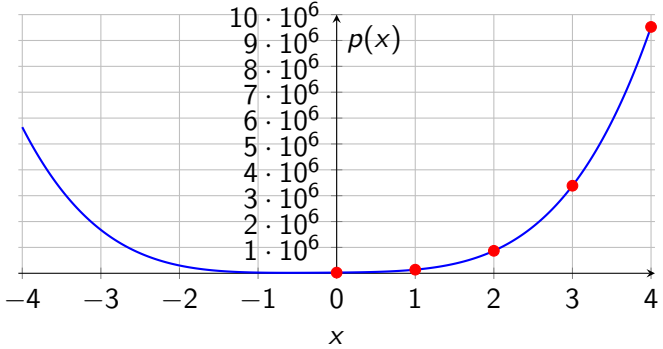
can uniquely be restored by any k distinct coordinates

$$(x_0, p(x_0)), \ldots, (x_k, p(x_k)).$$

Often choose $x_i = i$ for $i = 0, \ldots, k-1$: $\quad (0, p(0)), \ldots, (k, p(k))$
which are:

$$(0, 26725), (1, 139294), (2, 868585), (3, 3384706), (4, 9523717)$$

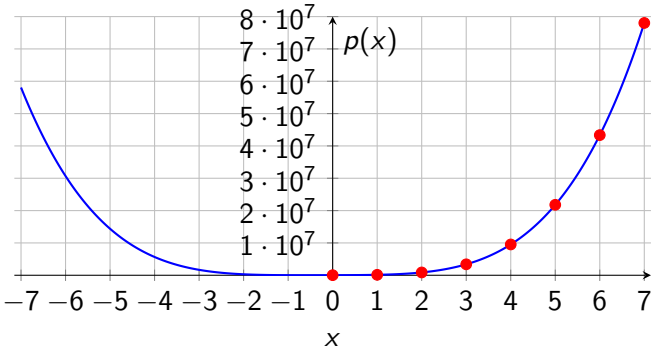
$$(0, 26725), (1, 139294), (2, 868585), (3, 3384706), (4, 9523717)$$



Redundancy

Don't just save k points of $p(x)$ but $n \geq k$:

$$(0, 26725), \quad (1, 139294), \quad (2, 868585), \quad (3, 3384706),$$
$$(4, 9523717), \quad (5, 21787630), \quad (6, 43344409), \quad (7, 78027970)$$

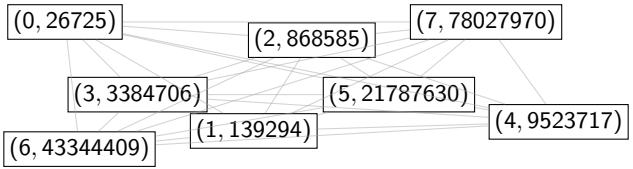


Redundancy

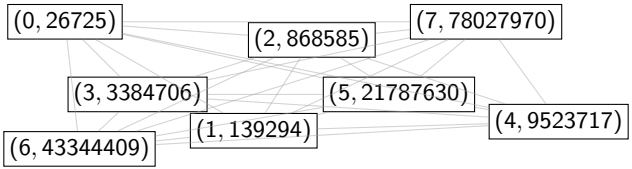
Don't just save k points of $p(x)$ but $n \geq k$:

$$(0, 26725), \quad (1, 139294), \quad (2, 868585), \quad (3, 3384706),$$
$$(4, 9523717), \quad (5, 21787630), \quad (6, 43344409), \quad (7, 78027970)$$

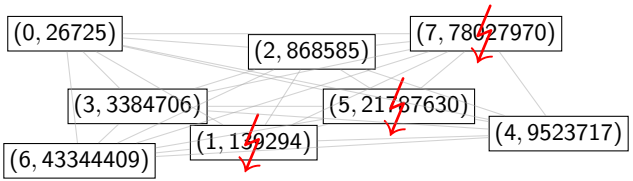
$\rightarrow$ Distribute n points over different disks?



Decoding



- Any $k = 5$ points $\rightarrow$ uniquely solvable system of linear equations



- Any $k = 5$ points $\rightarrow$ uniquely solvable system of linear equations

For instance, if nodes 1, 5 & 7 fail $\rightarrow$ recover from 0,2,3,4 & 6:

$$
\begin{aligned}
p(0) &= m_0 + 0 \qquad + 0 \qquad + 0 \qquad + 0 \qquad = 26725 \\
p(2) &= m_0 + m_1 \cdot 2 + m_2 \cdot 2^2 + m_3 \cdot 2^3 + m_4 \cdot 2^4 = 868585 \\
p(3) &= m_0 + m_1 \cdot 3 + m_2 \cdot 3^2 + m_3 \cdot 3^3 + m_4 \cdot 3^4 = 3384706 \\
p(4) &= m_0 + m_1 \cdot 4 + m_2 \cdot 4^2 + m_3 \cdot 4^3 + m_4 \cdot 4^4 = 9523717 \\
p(6) &= m_0 + m_1 \cdot 6 + m_2 \cdot 6^2 + m_3 \cdot 6^3 + m_4 \cdot 6^4 = 43344409
\end{aligned}
$$

$\rightarrow$ Recover the coefficients $m_0, \ldots, m_4$, i.e. the data

Problem (and why this was not 100% Erasure Coding)

Problem: for example, value

$$p(4) = 9523717 = (1001\,0001\,0101\,0010\,0000\,0101)_2$$

24 bits, instead of 16 bits.

Solution: work with Galois fields $GF(2^{16})$ (Reed-Solomon-Codes)

- Restrict to 16-bit chunks $\rightarrow$ integers $0, 1, \ldots, 2^{16} - 1$ for coefficients AND values $p(x)$
- Different computation ($\rightarrow$ see Galois fields)
- The result of uniqueness of polynomial still holds $\rightarrow$ uniquely recover data

Properties of Reed-Solomon-Codes 1/2

- Tolerate $n - k$ node failures
- Storage-optimal: can reconstruct data from any k out of n chunks with the minimum storage redundancy n/k
  - E.g. Google Colossus (9,6) redundancy 1.5,
  - And facebook f4 (14,10) redundancy 1.4
- Can choose any (n, k) (with $k \leq n$ and $n \leq 2^m$ with m the size of the chunks)

Properties of Reed-Solomon-Codes 2/2

Call the collection of n encoded chunks a *stripe*

- Scalable storage: split each of the k chunks into q chunks and encode the q stripes independently

- Can configure in *systematic form*: the n chunks are the k original chunks together with $n - k$ parity chunks

More Erasure Codes

- Overhead in repair and updates
- Regenerating codes (minimize repair bandwidth)
- Local Reconstruction codes (e.g. deployed by Microsoft Azure)
- SD-codes (disk and sector failures)
- RAID (array of disks)
- Low-density parity-check codes (SSDs)

...

Contact: gerlind@deschner.de

Thank you for your attention! Are there any questions?

References:

[1] Irving S. Reed and Gustave Solomon. "Polynomial Codes Over Certain Finite Fields". In: *Journal of the Society for Industrial and Applied Mathematics* 8 (1960), pp. 300–304. URL: https://sites.math.rutgers.edu/~zeilberg/akherim/reed.pdf (visited on 01/29/2026).

[2] Zhirong Shen et al. "A Survey of the Past, Present, and Future of Erasure Coding for Storage Systems". In: *ACM transactions on storage* 21 (2025), pp. 1–39. DOI: 10.1145/3708994.

[3] James Westall and James Martin. "An Introduction to Galois Fields and Reed-Solomon Coding". In: *School of Computing, Clemson University* (2010). URL: https://people.computing.clemson.edu/~westall/851/rs-code.pdf (visited on 01/29/2026).