Reed-Solomon

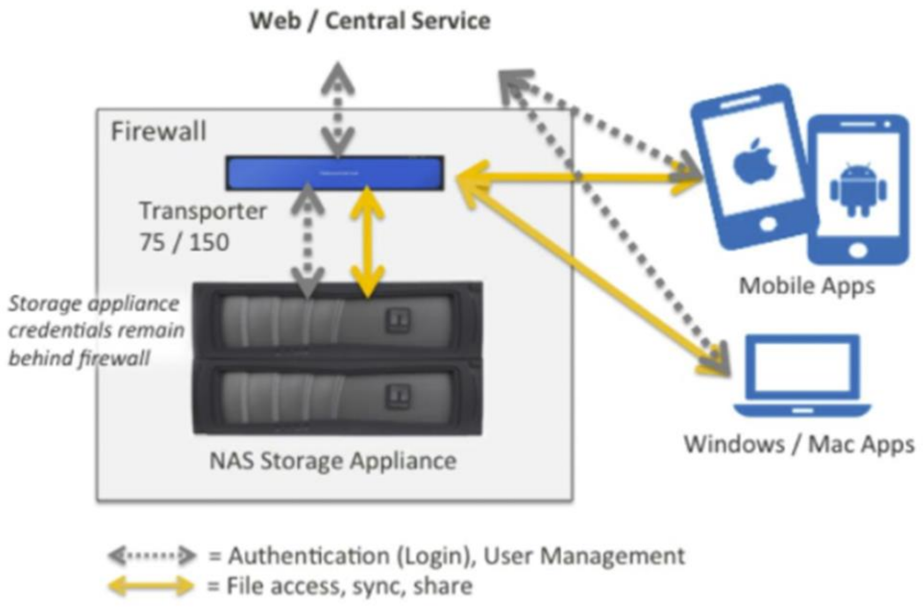
Smaller Companies Advancing On-Line Storage

Tom Coughlin, CONTRIBUTOR
I write about data storage **FULL BIO**
Opinions expressed by Forbes Contributors are their own.

Smaller companies can and must take greater risks than larger companies in order to create new market niches and grow their product offerings. Small innovative companies are a major source of advanced development and products that develop new markets and meet human needs. In this piece we will look at recent announcements by three such companies; Connected Data, Zadara and Backblaze.

After selling Drobo to an investor group, the founders of Connected Data have taken the transporter concept that they have applied to smaller NAS devices offering drop-box-like private cloud storage services and developed a larger private secure cloud storage offering for mid and large size businesses. Connected Data says that it has 40,000 active users and 23 PB of storage deployed so far for its Transporter business.

The company's Network Storage Connector is provided at no additional costs to the company's Transporter 75 and 150 private cloud storage appliances (the number refers to the number of organizational users). The Network Storage Connector syncs data stored on existing NAS systems with Transporter shared folders.

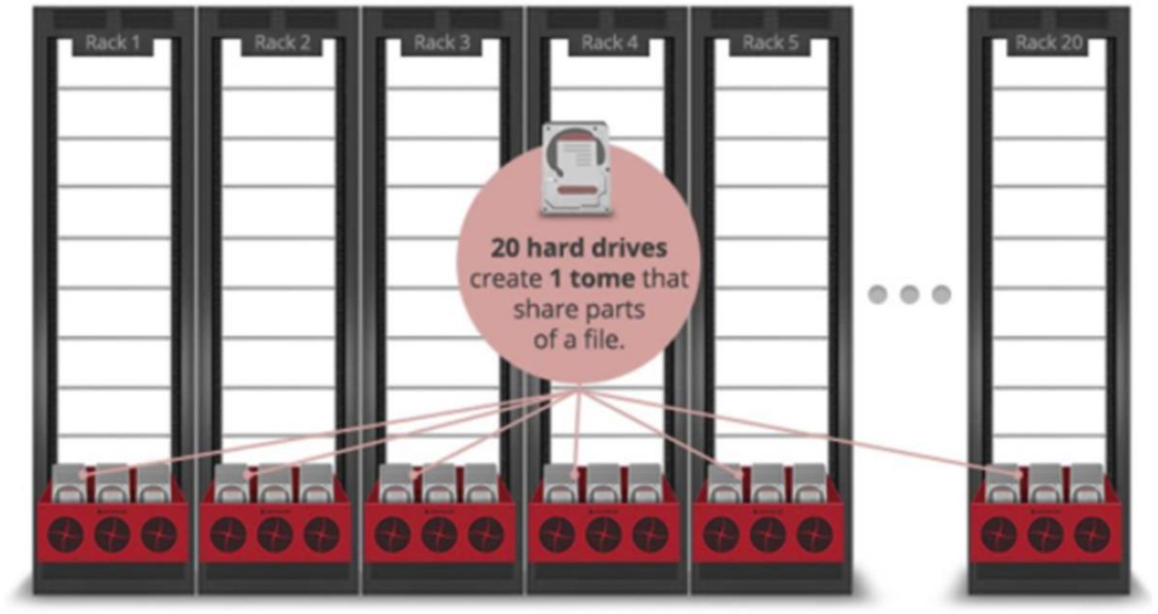

Web / Central Service
Firewall
Transporter
75 / 150
Storage appliance credentials remain behind firewall
NAS Storage Appliance
Mobile Apps
Windows / Mac Apps
= Authentication (Login), User Management
= File access, sync, share

The cloud storage appliances allow syncing important files and folders to laptops as well as mobile devices from any location. Terabytes of data can be accessed over the network while the actual data is only stored onsite in a user's facility. Keeping the data on-site assures regulatory compliance, privacy, security and control without resorting to a VPN. The new Transporter products support NetApp NTAP +0.12% Data ONTAP and Windows Server operating systems.

Zadara, a company offering a hybrid cloud storage service combining on-line storage with storage on site announced that it now has a Docker-based Zadara Container Service that allows running applications within the company's Virtual Private Storage Array (VPSA) rather than in compute servers. This approach allows very high performance and low latency as well as reduced traffic between application servers and storage.

Zadara also announced Zadara Backup to Amazon S3 (B2S3) that allows automatic snapshot-based continuous incremental backup from Zadara VPSA storage to low-cost Amazon S3 object storage. Data backed up to B2S3 can be restored to any storage medium from any vendor, including Amazon Elastic Block Storage (Amazon EBS). The company also announced that they are providing 800 GB SSD and 6 TB SATA hard disk drive storage options as part of their Storage as a Service offering.

On-line backup company Backblaze, which announced its Zetta-scale Backblaze Vaults in March, released its Reed-Solomon coding software library as open source. Reed-Solomon coding allows the sharding and reassembly of files in a storage system. Backblaze is well known for open sourcing its Storage Pod hardware as well as publishing reliability data for the hard disk drives it uses in its huge storage arrays.

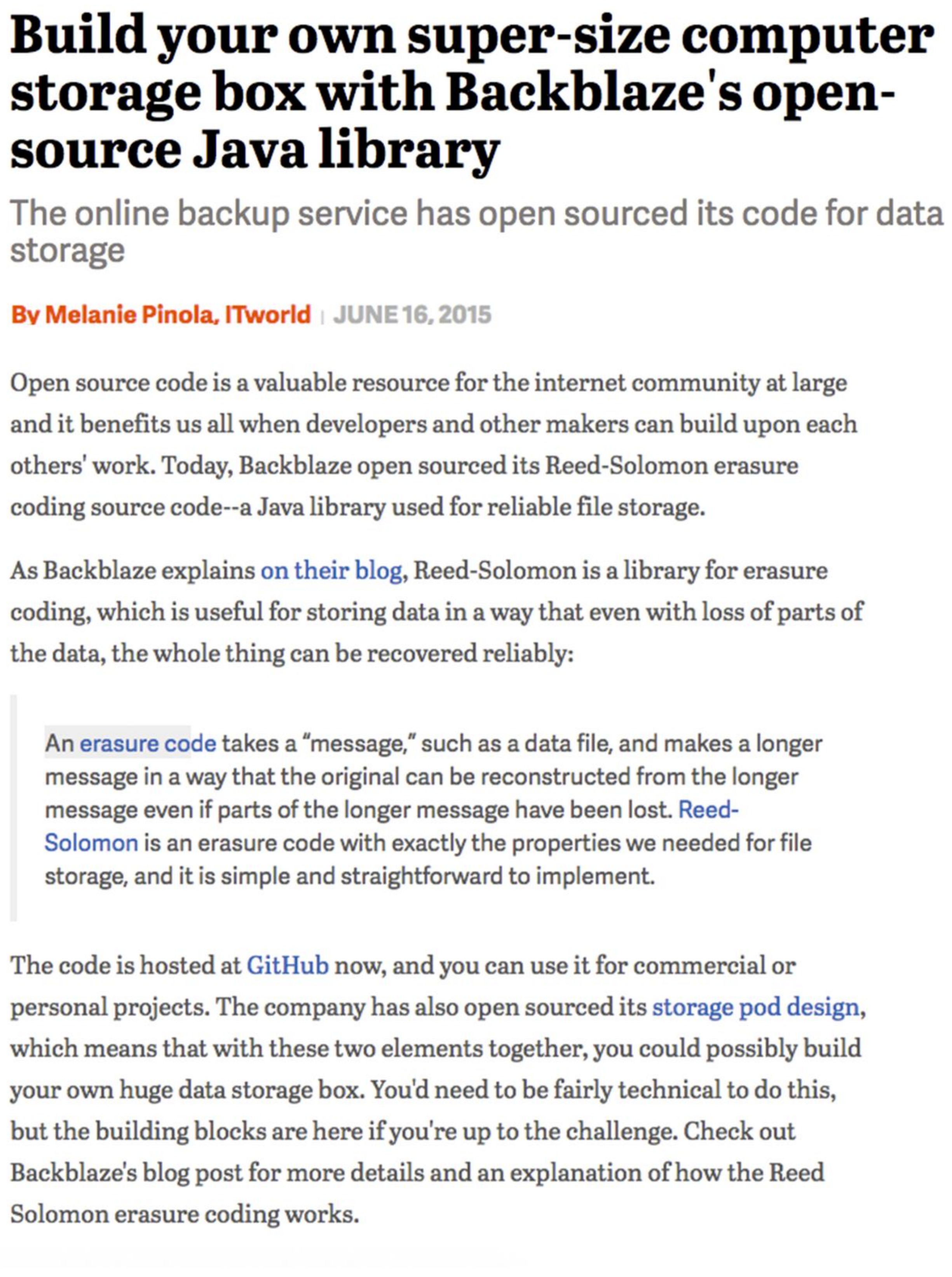


According to Backblaze CEO and founder, Gleb Budman, "Reed-Solomon erasure coding underlies the core systems behind Amazon S3, Linux RAID, and most of the mass storage systems and clouds. However, most of the implementation are proprietary, and there are no product quality Java versions that are open source." He went on to say that, "Not only is our Java version open source, but it is production quality (used by Backblaze in production, under massive load, at scale)...and it is as fast as implementations done in C (which surprised our team.)"

Small storage companies are creating new ways to store data and provide services that leverage existing hardware investments, provide novel hybrid cloud storage options and advance the development of open source hardware and software. These companies offer novel approaches to solving our storage needs and create new investment opportunities and advanced product services.

Tom Coughlin consults and writes on digital storage and applications. He is chairman of the Storage Visions, Creative Storage Conferences as well as the Flash Memory Summit, tomcoughlin.com

http://www.itworld.com/article/2936658/personal-technolo-
gy/build-your-own-super-size-computer-storage-box-with-backblazes-open-source-java-library.html



Build your own super-size computer storage box with Backblaze's open-source Java library

The online backup service has open sourced its code for data storage

By Melanie Pinola, ITworld | JUNE 16, 2015

Open source code is a valuable resource for the internet community at large and it benefits us all when developers and other makers can build upon each others' work. Today, Backblaze open sourced its Reed-Solomon erasure coding source code--a Java library used for reliable file storage.

As Backblaze explains on their blog, Reed-Solomon is a library for erasure coding, which is useful for storing data in a way that even with loss of parts of the data, the whole thing can be recovered reliably:

> An erasure code takes a "message," such as a data file, and makes a longer message in a way that the original can be reconstructed from the longer message even if parts of the longer message have been lost. Reed-Solomon is an erasure code with exactly the properties we needed for file storage, and it is simple and straightforward to implement.

The code is hosted at GitHub now, and you can use it for commercial or personal projects. The company has also open sourced its storage pod design, which means that with these two elements together, you could possibly build your own huge data storage box. You'd need to be fairly technical to do this, but the building blocks are here if you're up to the challenge. Check out Backblaze's blog post for more details and an explanation of how the Reed Solomon erasure coding works.

https://www.extremetech.com/computing/207912-se-
crets-of-the-vault-backblaze-open-sources-key-sections-of-its-data-preservation-software

Secrets of the vault: Backblaze open-sources key sections of its data preservation software

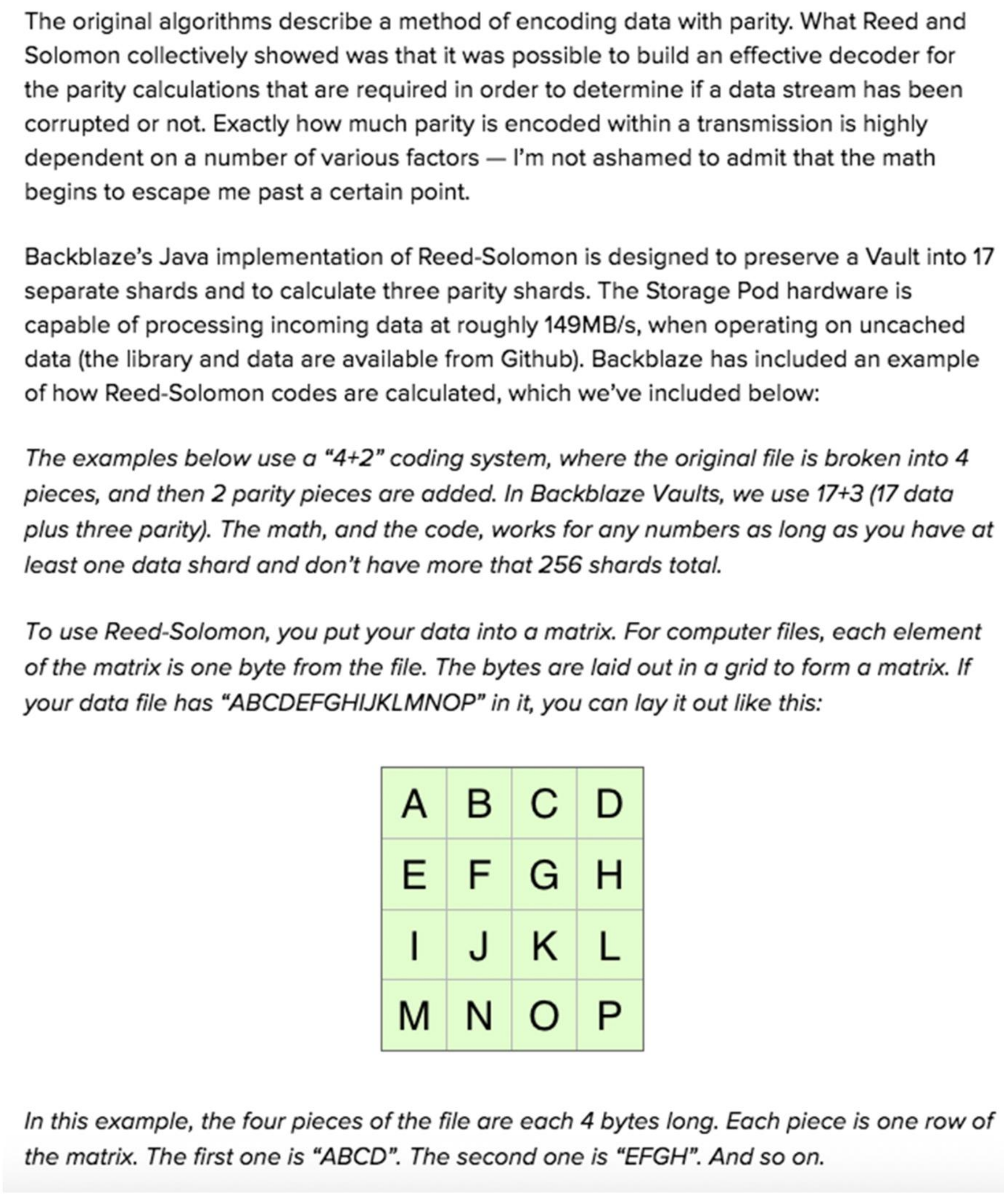
By Joel Hruska on June 16, 2015 at 11:00 am

Of all the various backup companies on the market, few have documented their work and research as thoroughly as Backblaze. The company has previously made headlines for open-sourcing both the underlying hardware design that it uses for its Storage Pods and its hard drive reliability data (the latter early this year). Now, Backblaze is opening up another facet of its operation — the implementation of its Reed-Solomon error-correcting codes.

Reed-what?

Reed-Solomon error correcting codes are a critically important underpinning of computing that you've probably never heard of. First created by Irving S. Reed and Gustave Solomon in 1960, they form the basis of error correction as it's used in a huge number of products, including CDs, DVDs, Blu-rays, QR Codes, multiple data transmission standards, broadcast television standards, and RAID 6 itself. Not every parity system in the world directly relies on Reed-Solomon, but the ideas behind Reed-Solomon are the basis for error correction.

https://www.extremetech.com/computing/207912-se-
crets-of-the-vault-backblaze-open-sources-key-sections-of-its-data-preservation-software



The original algorithms describe a method of encoding data with parity. What Reed and Solomon collectively showed was that it was possible to build an effective decoder for the parity calculations that are required in order to determine if a data stream has been corrupted or not. Exactly how much parity is encoded within a transmission is highly dependent on a number of various factors — I'm not ashamed to admit that the math begins to escape me past a certain point.

Backblaze's Java implementation of Reed-Solomon is designed to preserve a Vault into 17 separate shards and to calculate three parity shards. The Storage Pod hardware is capable of processing incoming data at roughly 149MB/s, when operating on uncached data (the library and data are available from Github). Backblaze has included an example of how Reed-Solomon codes are calculated, which we've included below:

The examples below use a "4+2" coding system, where the original file is broken into 4 pieces, and then 2 parity pieces are added. In Backblaze Vaults, we use 17+3 (17 data plus three parity). The math, and the code, works for any numbers as long as you have at least one data shard and don't have more that 256 shards total.

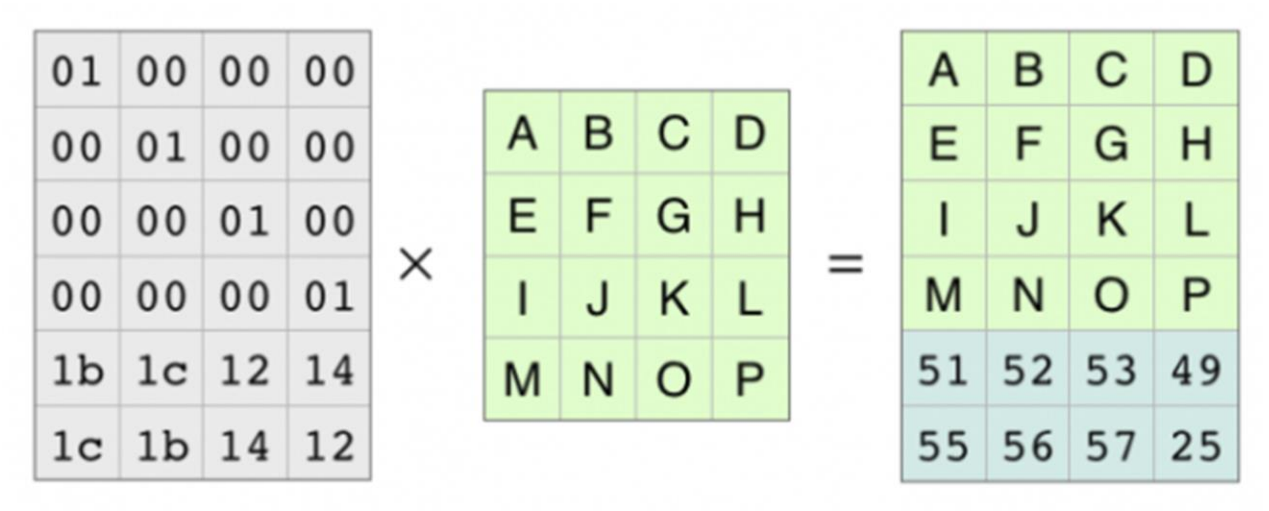
To use Reed-Solomon, you put your data into a matrix. For computer files, each element of the matrix is one byte from the file. The bytes are laid out in a grid to form a matrix. If your data file has "ABCDEFGHIJKLMNOP" in it, you can lay it out like this:

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

In this example, the four pieces of the file are each 4 bytes long. Each piece is one row of the matrix. The first one is "ABCD". The second one is "EFGH". And so on.

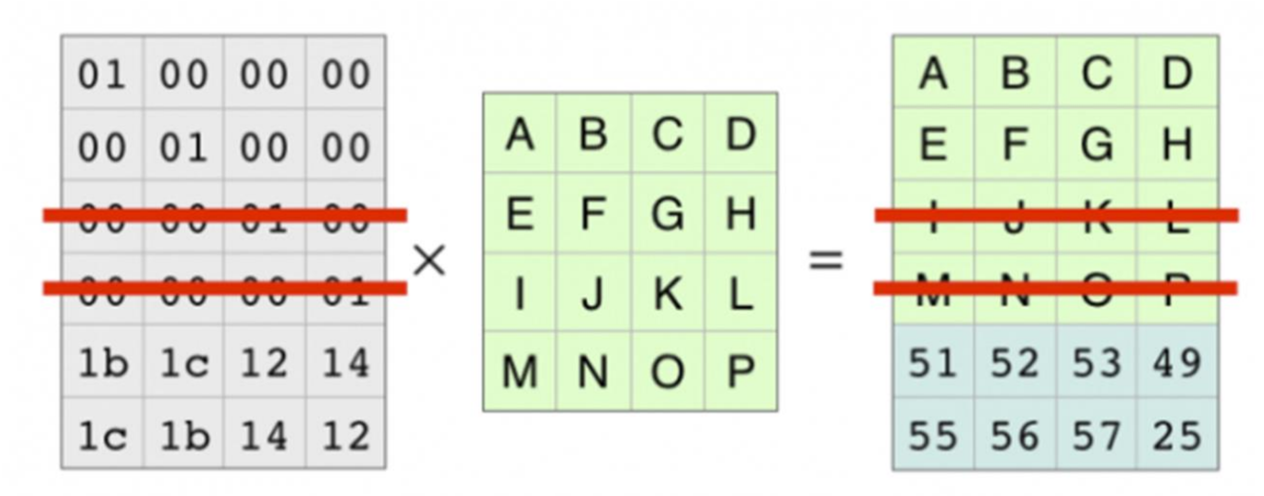
https://www.extremetech.com/computing/207912-secrets-of-the-vault-backblaze-open-sources-key-sections-of-its-data-preservation-software

The Reed-Solomon algorithm creates a coding matrix that you multiply with your data matrix to create the coded data. The matrix is set up so that the first four rows of the result are the same as the first four rows of the input. That means that the data is left intact, and all it's really doing is computing the parity.



The result is a matrix with two more rows than the original. Those two rows are the parity pieces.

Each row of the coding matrix produces one row of the result. So each row of the coding matrix makes one of the resulting pieces of the file. Because the rows are independent, you can cross out two of the rows and the equation still holds.



https://www.extremetech.com/computing/207912-secrets-of-the-vault-backblaze-open-sources-key-sections-of-its-data-preservation-software

And with those rows completely gone it looks like this:

$$\begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 1b & 1c & 12 & 14 \\ 1c & 1b & 14 & 12 \end{bmatrix} \times \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ 51 & 52 & 53 & 49 \\ 55 & 56 & 57 & 25 \end{bmatrix}$$

Because of all the work that mathematicians have done over the years, we know the the coding matrix, the matrix on the left, is invertible. There is an inverse matrix that, when multiplied by the coding matrix produces the identity matrix. As in normal algebra, in matrix algebra you can multiply both sides of an equation by the same thing. In this case, we'll multiply on the left by the identity matrix:

$$\begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 8d & f6 & 7b & 01 \\ f6 & 8d & 01 & 7b \end{bmatrix} \times \begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 1b & 1c & 12 & 14 \\ 1c & 1b & 14 & 12 \end{bmatrix} \times \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix} = \begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 8d & f6 & 7b & 01 \\ f6 & 8d & 01 & 7b \end{bmatrix} \times \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ 51 & 52 & 53 & 49 \\ 55 & 56 & 57 & 25 \end{bmatrix}$$

The inverse matrix and the coding matrix cancel out.

$$\cancel{\begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 8d & f6 & 7b & 01 \\ f6 & 8d & 01 & 7b \end{bmatrix}} \times \cancel{\begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 1b & 1c & 12 & 14 \\ 1c & 1b & 14 & 12 \end{bmatrix}} \times \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix} = \begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 8d & f6 & 7b & 01 \\ f6 & 8d & 01 & 7b \end{bmatrix} \times \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ 51 & 52 & 53 & 49 \\ 55 & 56 & 57 & 25 \end{bmatrix}$$

Which leaves the equation for reconstructing the original data from the pieces that are available:

$$\begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{bmatrix} = \begin{bmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 8d & f6 & 7b & 01 \\ f6 & 8d & 01 & 7b \end{bmatrix} \times \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ 51 & 52 & 53 & 49 \\ 55 & 56 & 57 & 25 \end{bmatrix}$$

So, to make a decoding matrix, the process is to take the original coding matrix, cross out the rows for the missing pieces, and then find the inverse matrix. You can then multiply the inverse matrix and the pieces that are available to reconstruct the original data.

Backblaze hasn't clarified exactly why they're open-sourcing this section of their code, but presumably it's to offer useful data and inspiration to individuals looking to roll their own solutions — or just to get eyes and fingers on the codebase in the hopes of improving it further.