纠删码相对于之前多副本的区别：

存储空间降低 ：原来3副本需要3倍空间存储一份数据，纠删码只需要1.4倍空间存储一份数据。

可靠性：3副本允许坏的副本数：2/3； 纠删码允许坏的副本数：4/14

额外负担：

计算量（只要有一个坏盘就得通过网络读出n倍的数据并重新计算）

数倍的网络负载

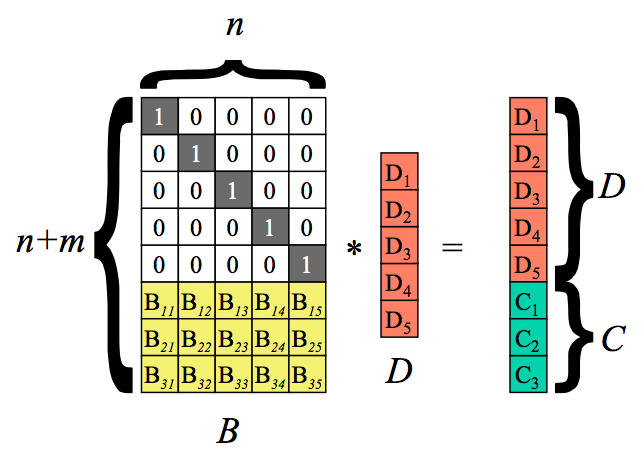
2.  算法核心

下面重点讲一讲Reed-Solomon(RS)码：

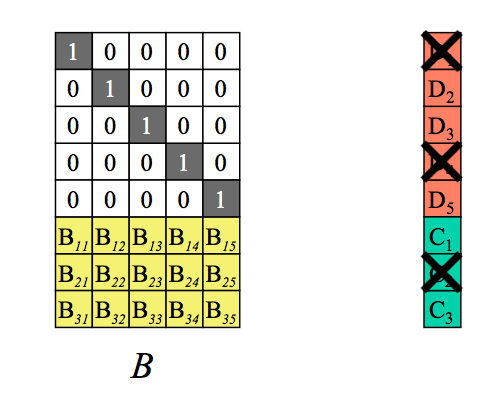
Reed-Solomon（RS）码是存储系统较为常用的一种纠删码，它有两个参数n和m，记为RS(n,m)。n代表原始数据块个数。m代表校验块个数。接下来介绍RS码的原理。

RS码原理

以n=5，m=3为例。即5个原始数据块，乘上一个(n+m)\*n的矩阵，然后得出一个(n+m)\*1的矩阵。根据矩阵特点可以得知结果矩阵中前面5个值与原来的5个数据块的值相等，而最后3个则是计算出来的校验块。

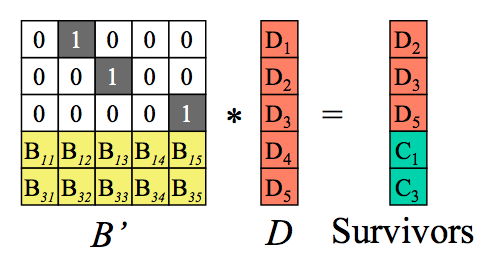


以上过程为编码过程。D是原始数据块，得到的C为校验块。假设丢失了m块数据。如下：

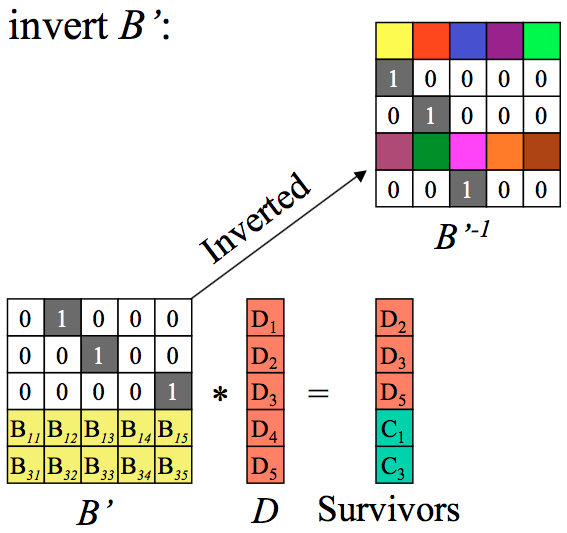


那我们如何从剩余的n个数据块（注意，这里剩余的n块可能包含几个原始数据块+几个校验块）恢复出来原始的n个数据块呢，就需要通过下面的decoding（解码）过程来实现。

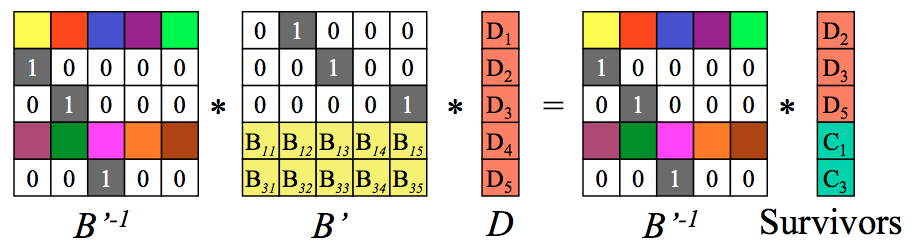
第一步：从编码矩阵中删去丢失数据块和丢失编码块对应行。 将删掉m个块的(n+m)\*1个矩阵变形为n\*1矩阵，同时B矩阵也需要删掉对应的m个行得出一个B'的变形矩阵，这个B'就是n\*n矩阵。如下：假设D1、D4、C2丢失，我们得到如下B’矩阵及等式。



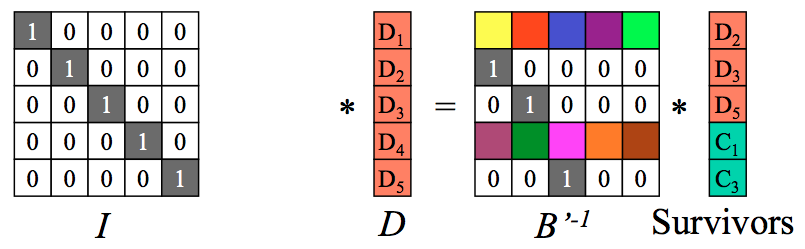
第二步：求出B’的逆矩阵。



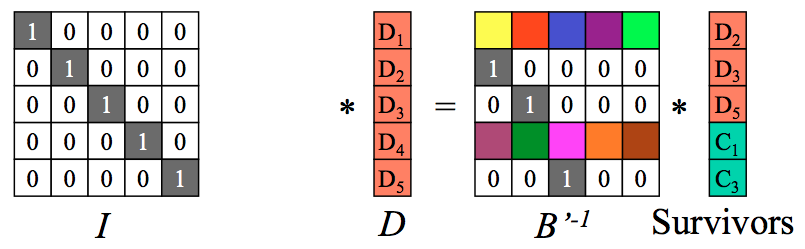
第三步：等式两边分别乘上B’的逆矩阵。



B’和它的逆矩阵相乘得到单位矩阵I，如下：



左边只剩下原始数据矩阵D：



至此完成解码过程。

注：图中黄色部分为范德蒙矩阵。至于如何生成B矩阵，以及如何求B’的逆矩阵，请自行百度。