**1 Introduction & Motivation**

在大数据时代，数据的拥有者对数据的机密性和可靠性提出了越来越高的要求。首先，高的机密性是至关重要的。近些年来，各种黑客的攻击手段越来越高明，攻击方式越来越多样，恶性攻击事件层出不穷。例如，2017年美国最大的信用评级机构Equifax遭到黑客入侵，导致1.47亿用户个人敏感数据泄露；2020年芬兰心理治疗机构Vastaamo遭到黑客攻击，数万患者病历数据泄露[1]。其次，高的可靠性同样重要。比如，911袭击事件之后，纽约银行和德意志银行的数据中心因此毁于一旦。6个月后，纽约银行破产清盘， 而德意志银行因为在几十公里外做了数据备份，得以存活。

黑客攻击的最常用手段是窃取和破坏。防范窃取主要依靠加密，比如微软公司开发的一种基于NTFS文件系统的保护数据机密性的技术EFS，就极大提高了数据的机密性[2]；防范破坏，也就是提高可靠性，主要依靠冗余，一般是副本[3]和纠删码[4]。RAID技术是使用最广泛的纠删码技术，其通过存储奇偶校验信息的方式，实现对数据的冗余保护。想要同时兼顾数据的可靠性和机密性，往往需要同时利用纠删码技术和加密算法，大致分为两种思路：一种是传统的加密方法与纠删码技术的结合使用，不论是先加密再做校验计算，还是先做校验计算再加密，都会严重影响计算性能，提升数据的空间占用；另一种是为了提高性能，将纠删码与不基于加密的安全机制结合，但机密性又受到了影响。本文为了解决这个问题，提出一种新的思路，即将加密算法与纠删码融合到一起，赋予纠删码机密性，同时保障了数据的可靠性、机密性以及性能。

本文的主要贡献包括：

1）本文提出了DERAID，一种融合了加密算法，兼备机密性、可靠性和高性能的RAID技术。将密钥与本文提出的检验块结合取代了校验块的作用，将ECB加密算法[5]与阵列纠删码技术结合，使得编码等过程的算法的选取利用更加灵活，且保持了较高的机密性和可靠性。

2）本文提升了同时满足可靠性和机密性条件的计算性能。通过能够使算法选取更加灵活的DERAID5阵列，大幅提升了编码、更新、重建等过程的性能。测试结果表明：DERAID5在保持机密性与可靠性的前提下，与传统方法、QS-code相比，编码过程性能提高44.7%~95.3%，更新过程性能提高48.2%~99.6%，重建过程性能提高63.0%~99.9%。

3）本文降低了同时满足可靠性和机密性条件的数据膨胀率。最高数据膨胀率可降低3.3%~36.9%。

本文的各章节安排为：第一章是引言与动机，第二章是相关工作，第三章是对DERAID技术的介绍，包括具体的编码方式、可靠性与机密性的分析，第四章是实验对比，包括RAIDM、MRAID、DERAID5和QS-code编码、解码、更新、重建过程的性能对比分析与膨胀率对比，第五章是总结。

**2 Related Work**

面对兼顾数据的可靠性和机密性的需求，不论是学术界还是工业界大多都是独立研究纠删码和加密，极少相关工作是将两者相结合研究。

对于对纠删码的研究，很多研究人员取得了不小的进展。比如Dimakis等人提出了再生码[6]的概念。这是一种新型的纠删码，允许新节点通过与存活节点的通信来获取存储数据的函数，而不是重建整个编码数据对象。这种方法不仅可以显著减少修复带宽的需求，还极大优化了重建过程。Gopalan, P.等人提出了局部修复码[7]，他们在论文首次定义了纠删码中坐标的局部性概念，通过优化局部性和冗余度，提高了存储效率和数据恢复速度。而与局部修复码相似的分组纠删码[8]的概念是由Sathiamoorthy等人提出的，他们提出的这种纠删码将数据块分成若干组，每组产生一个局部编码块，通过在局部性和最小距离之间找到最优的权衡，实现了比传统的Reed-Solomon码更高效的修复性能和更高的可靠性。随着近些年来RAID技术[9]的发展，已经产生了多种不同的RAID级别，每种级别都有其特定的优势和适用场景[10]。RAID技术发展至今，形成了RAID0,RAID1[3],RAID4,RAID5，RAID6等具有良好扩展性的多个级别。许多人提出了不同的扩展方案用来提升RAID技术的性能。比如Guangyan Zhang等人提出了一种新的数据重新分配方法McPod来加速RAID-4扩展，该方法在保持所有数据磁盘上数据均匀分布的同时，最大限度地减少了要移动的数据块数量[11]。LIANG J等人提出了一种创新的RAID-5扩展方法，通过减少数据迁移量和避免奇偶校验块的重新计算，显著降低了扩展时间和成本[12]。Ziyang Jiao等人提出了Asymmetric RAID，它是一种针对SSD异构性而设计的新型RAID架构，它通过不对称地分配数据和差异化地导出地址空间来改善性能和存储利用率，可以应用于各种需要高性能和高可靠性的存储场景[13]。

对加密的研究，许多研究人员也有突破。比如，在对对称密码学的分组密码的研究中，Pierre Junod等人曾经提出了一个新的块密码家族FOX[14]，以应对当时趋势中的块密码需求，同时避免使用轻量级密钥调度和纯代数构造的S盒。FOX的设计还提供了最大的灵活性，包括轮数、密钥大小、块大小以及实现问题，并能在硬件上与最快的块密码性能相媲美。还有在对称密码学的流密码与分组密码结合的研究中，Hassan Noura等人提出混沌加密方案结合输出反馈模式（OFB Mode）[15]，将混沌加密方案与分组密码的OFB模式结合，提高了加密过程的效率和安全性。

为了保障纠删码的机密性，工业界大都是将纠删码与加密算法结合使用。RAID5与加密结合方面，比较有代表性的，比如在 Linux 内核[16]中，RAID5 的冗余和容错机制通过 raid5.c 文件实现，该文件利用 异或（XOR）算法 来计算和恢复数据块的校验信息，从而提供数据冗余。与此同时，dm-crypt.c 文件负责加密块设备上的数据，它采用 AES 等对称加密算法，结合 ECB（Electronic Codebook） 加密模式对磁盘数据进行加密和解密处理。ECB 模式将每个数据块独立加密，确保数据的机密性。这两个模块的结合有效地实现了冗余保护与加密保障的双重目标。还有ZFS的RAID-Z功能及其集成加密模块Zcrypt[17]，vdev\_raidz.c 文件在 OpenZFS 中实现了类似于纠删码的冗余机制，通过数据校验和分布式存储保证了数据的可靠性和容错能力。同时，dsl\_crypt.c 文件则提供了加密机制，管理加密密钥并加密数据集，确保了数据在存储过程中的机密性，从而在系统中提供了双重保障，既保证了数据的可靠性，也维护了其机密性。而RAID6与加密结合方面，就要说到TrueNAS[18]。它是一个基于 **OpenZFS** 的开源存储操作系统，提供强大的存储、管理和数据保护功能。TrueNAS 支持 **RAID-Z2**，这是 **RAID-Z** 的增强版，能够通过双重校验（parity）提供与 **RAID 6** 类似的数据冗余和高可用性，确保数据在两个硬盘故障的情况下依然保持完整。除了冗余保护，TrueNAS 还支持内置的 **AES 加密**，为存储池中的数据提供加密保护，确保数据的机密性和安全性。然而，虽然这种简单的叠加结合使用早已很成熟，但是巨大的计算消耗是不可忽视的。

除了利用加密方案保障纠删码的机密性，一些研究人员也曾经提出了不少不基于加密体制的纠删码数据机密性的保障方案。比如Yamamoto等人提出的基于秘密共享[19]的门限存储机制[20],他们描述了如何使用(k,L,n)阈值方案构建一个秘密分享系统。该系统能够将一个秘密分割成n个份额，并分配给n个参与者。只有当至少k个参与者提供他们的份额时，才能恢复出原始的秘密。为了防止参与者之间的合谋，它们还提出了一种额外的机制。当参与者数量超过k但未达到L时，他们无法恢复出原始的秘密。这种机制通过增加额外的份额数量L来实现，从而在某种程度上防止了合谋的发生。再有Pawar S.等人提出了安全纠删码[21]方案，其主要关注云计算数据分别存储在不同节点中,面临部分节点及通信链路被窃听的风险。随后，涌现了安全纠删码方案[22-23]。Wang C Z. K.等人提出了QS-Code[24]，其作为一种具有部分安全性的准系统纠删码，在云存储领域具有广泛的应用前景和重要的学术价值。它的准系统性、部分安全性、纠删码特性、高效性和适应性等特点使得它能够有效地应对云存储环境中的各种挑战，提高数据的可靠性和安全性。然而，虽然QS-Code提供了部分安全性，但它可能不足以满足对数据安全性要求极高的应用场景。而且尽管 QS-Code 设计目标是提高效率，但在实际应用中，某些情况下其编码和解码过程可能会比较复杂，增加了计算和存储的开销。QS-code作为与本文相关性最高的文献，将在实验部分重点进行比较。

**3 Methodology**

DERAID5与RAID5一样数据以块来存储。一般来说，RAID5一个条带五个条块，但实际不限制行数列数，DERAID5也是一样，所以我们在本章只取其中一个条带来分析。我们把条带分成K个条块。则我们选取的一个条带有K列。

与之前RAID各个级别不同的是，DERAID是将本文中提出的检验块与密钥结合起到校验块的作用，通过此使得优化的ECB加密体制与阵列纠删码技术结合到了一起，既利用加密体制保证了阵列的机密性，又通过纠删码技术保证了阵列的可靠性，同时避免了传统方法要单独加密所造成的巨大计算开销。本文是在使用最为广泛的RAID5中进行了加密集成。

**3.1 DERAID6（最高容忍二故障）**

1. **编码过程（写）**

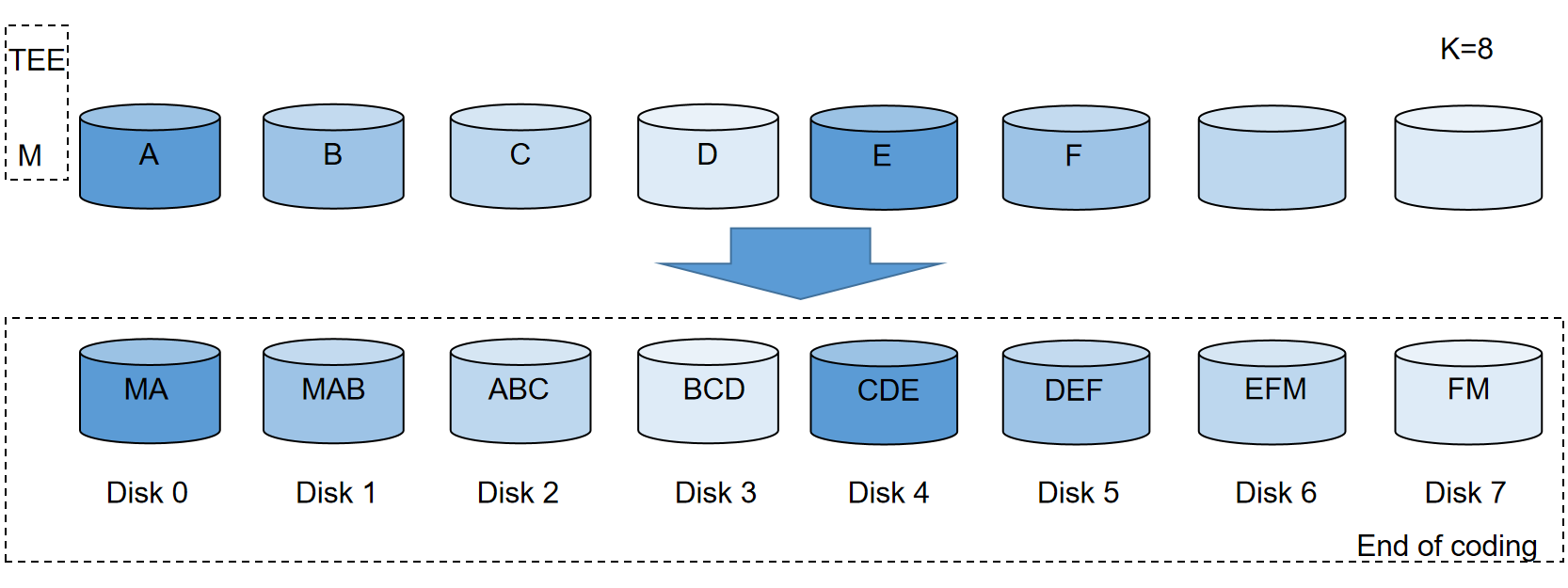
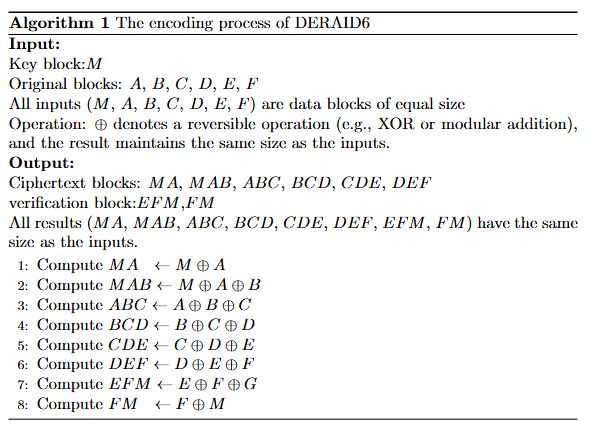


FIGURE 1. DERAID6编码过程

编码过程如图FIGURE 1，是一个条带中8个条块存储的6个原始数据块编码为8个密文块的过程。EFM密文块和FM密文块是该条带的冗余数据，而密钥块M则相当于各个条带共同的冗余数据，M的冗余基本可以忽略不计。本文的其他过程图同样采取该图的结构样式。

整个编码过程中，整体上遵从每个密文块都是由当前对应的原始数据块及前两个数据块运算得到的原则，例如第三个密文块由A,B,C共同运算得到，这里选取的运算只要能满足AB与A，B的数据量相同且能进行逆运算即可。但是也有例外，第一个密文块由密钥块M与A进行运算得到MA。倒数第二个密文块，也是检验块，由前两个原始数据块E，F以及M共同运算得到。倒数第一个密文块，同样也是检验块，由原始数据块F以及M运算得到。根据以上完成该示例条带的编码过程，如图FIGURE 1所示。

这里是，编码过程的具体算法描述：



更一般的说，假设密钥块为M，一条条带中的条块数位n，从编码起始到结束每一个条块为Ni（1≤i≤n,i∈N），而每一个条块的原始数据为Ni-p，每一个条块的编码后数据为Ni-c。

（1）

例如，若M=0x209，N1-p=0x26d，N2-p=0x30a，

当i=1，则N1-c=M⊕N1-p=0x209⊕0x26d=0x64.

当i=2，则N2-c=M⊕N2-p⊕N1-p=0x209⊕0x30a⊕0x26d=0x36e.

后依此类推。

上述为示例条带编码过程，我们将该过程扩展到多个条带。同时DERAID6仿照RAID 6实现负载均衡的做法，将数据和检验信息以条带的形式均匀分布在所有硬盘上，每个硬盘都存储部分数据和奇偶校验信息。这样在读写数据时，可以同时对多个硬盘进行操作，避免了单一硬盘的负载过重，从而实现负载均衡。

根据以上，得到的最终的多条带的结果，方框框起来的是检验块，如图FIGURE 2.

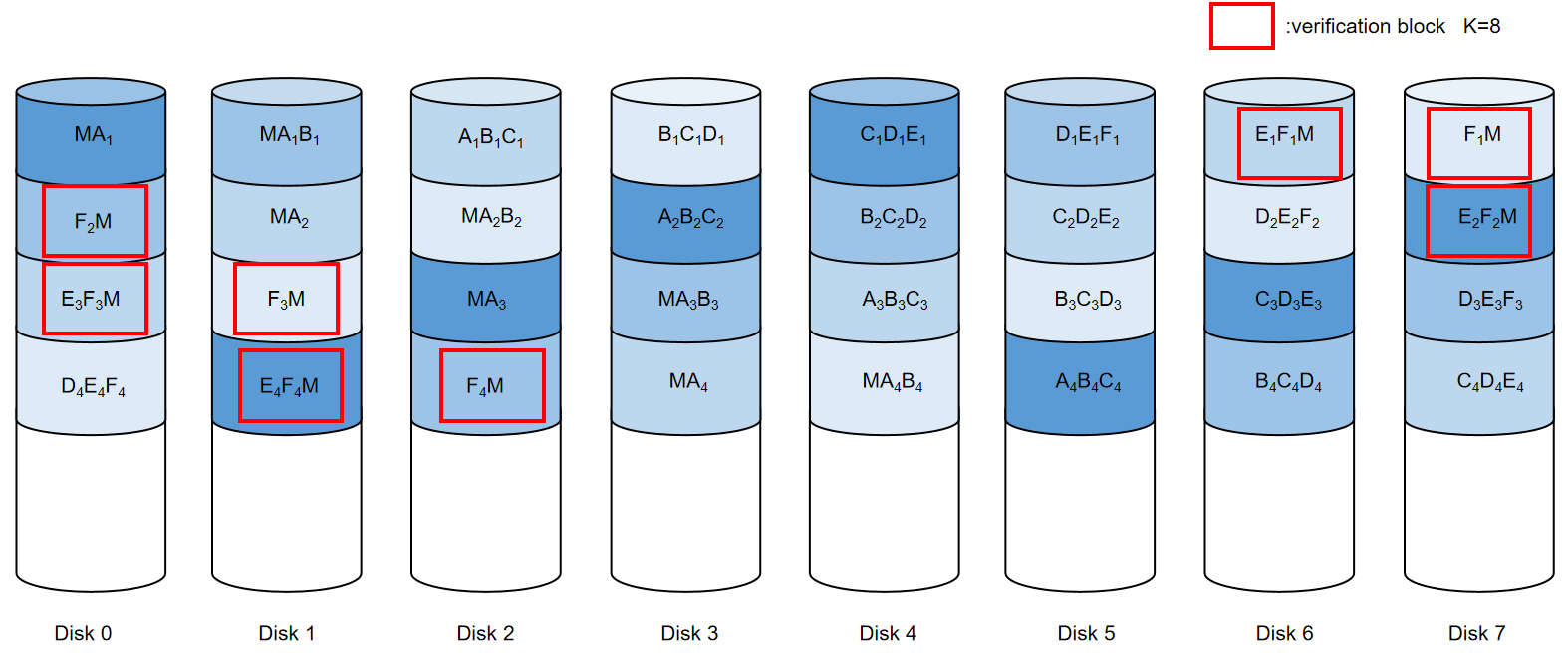


FIGURE 2. DERAID5编码结束扩展

1. **解码过程（读）**
2. **更新过程（查）**
3. **检验过程**
4. **重建过程**

**假如disk2与disk3被破坏或丢失，即ABC、BCD密文块丢失，以示例条带为例。重建过程如下：**

|  |  |
| --- | --- |
| **正方向恢复数据** | **逆方向恢复数据** |
| **M⊕MA=A** | **MF⊕M=F** |
| **M⊕A⊕MAB=B** | **MEF⊕F⊕M=E** |
|  | **DEF⊕E⊕F=D** |
|  | **CDE⊕D⊕E=C** |

**3.2可配比例DERAID**

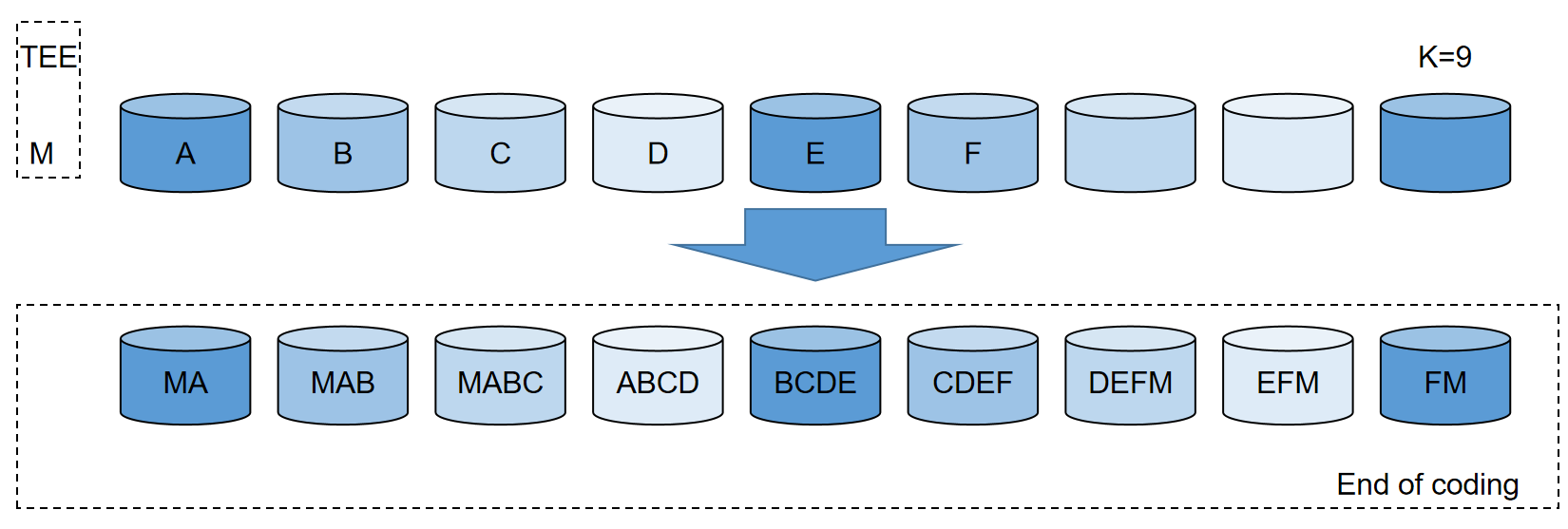


FIGURE 2. DERAID（6+3）编码过程

通过增加原始数据块重复异或的次数，使得包含同一个原始数据块的密文块的数量增加，同时增加冗余数据块数量，从而实现容忍更多故障的方案。

以容忍三故障的可配比例DERAID为例，每一个原始数据块重复利用四次，且检验块数量增加到三个。

如原始数据块A，分别保存到MAB、MABC、ABCD、BCDE四个密文块中，如果发生三故障，且都包含A，那么可通过另外一个包含A的密文块与其他密文块计算恢复A。其他原始数据块同样。

与传统方法相比没有增加额外存储空间，且保证了机密性，可靠性，与DERAID5一样具有计算性能优势。唯一与DERAID5相比不足的是，有少部分三故障情况无法重建，可靠性相比传统方法大致降低0.04%。能容忍更多数量故障的方案同样，可靠性会稍微降低。

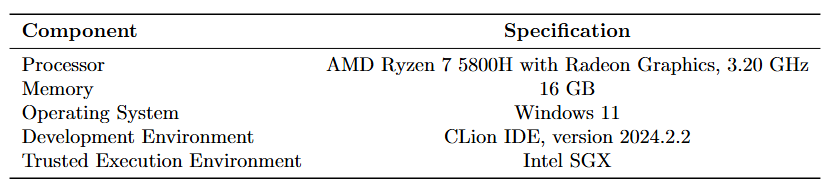
**4 Experiment**

**4.1 实验配置**

因为XOR运算其独特的性质、广泛的应用场景以及在处理大规模数据时的高效性和准确性，许多实验都基于XOR运算[28]，比如一些专门为了提高性能的基于XOR运算的RAID6扩展方法[29]。再比如J. Wu等人提出的CLEC方法中，每个节点对包含数据块的延迟圈中的每个 chunk 都是进行的 xor 运算，以计算组合的 chunk；而且每个块的组合也只需要xor操作[30]。所以这里我们也基于XOR运算来进行性能测试。

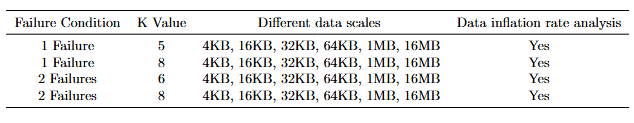
本实验的配置如表TABLE 3所示。

TABLE 3. System Configuration for Experimental Setup



在实验比较中，我将DERAID5与“先RAID5后加密”（简写为RAID5M) [16、17]、“先加密后RAID5”(简写为MRAID5) [16、17] 、QS-code代码 [24] 进行比较，将DERAID5（2）与“先RAID6后加密”（简写为RAID6M) [18]、“先加密后RAID6”(简写为MRAID6) [18] 、QS-code代码 [24] 进行比较。都分别采用4KB、16KB、32KB、64KB、1MB、16MB的数据规模，在不同的比例情况下，本文对上述方法的编码过程、解码过程、更新过程、重建过程四个过程进行了对比实验，最终收集到的用于比较的实验的数据都是重复十次，去掉最高值和最低值之后的平均值。同时我们做了相应的数据膨胀率分析实验。如表TABLE 4。在下面的实验中故障硬盘数量假设为G。本文的代码已开源：<https://github.com/TLMOA/DERAID。>

TABLE 4. Comparison of different experiments



3.实验方案

|  |  |  |
| --- | --- | --- |
|  | 不同比例 | 不同数据规模 |
| DERAID6与raid6+加密比较 | 4+2、6+2 | 4KB、16KB、32KB、64KB、1MB、16MB |
| DERAID6与SEC方案比较 | 4+2、6+2 | 4KB、16KB、32KB、64KB、1MB、16MB |
| DERAID(3)与SEC方案比较 | 6+3、8+3 | 4KB、16KB、32KB、64KB、1MB、16MB |