面向大容量数据实时传输的块间 4 纠删编码

陈钢[1]，朱俊峰[1]，张世乐[2]，吴百锋[1]
(1. 复旦大学 计算机科学技术学院，上海 201203；2. 诺基亚西门子通信有限公司，浙江 杭州 310005)

摘 要：现有纠删编码由于编解码运算复杂度及编码数据大小的限制，很难适用于面向大容量数据块或数据分组的实时传输中。在奇偶校验码的基础上，结合有限域 GF(2^8)域的特性，提出一种新的面向大容量数据块实时传输的块间 4 纠删编码方案。该编码方案对一组连续的数据块使用 4 个冗余数据块，即可容忍任意 4 个数据块同时差错。编解码运算的复杂度与数据块的大小成线性关系，对数据块的大小没有限制。编码具有 MDS 性质，在同等冗余条件下达到最佳的纠删能力。理论和实验分析表明：该编码方案最大能够对连续 27 个数据块进行块间纠删编码。

关键词：纠删编码；大容量数据；奇偶校验码；GF(2^8)域；MDS
中图分类号：TP333 **文献标识码**：A **文章编号**：1000-436X(2012)06-0040-10

Inter-block 4-erasure-correcting coding scheme for real-time bulk data transfer

CHEN Gang[1], ZHU Jun-feng[1], ZHANG Shi-le[2], WU Bai-feng[1]
(1. School of Computer Science and Technology, Fudan University, Shanghai 201203, China;
2. Nokia Siemens Networks, Hangzhou 310005, China)

Abstract: Most of the existing erasure-correcting codes were limited by encoding/decoding complexities and encoded data size. They were not suitable for use in the real-time applications that orient bulk data based on blocks or packages. Therefore, a novel inter-block 4-erasure-correcting coding scheme for real-time bulk data transfer was presented. Based on single parity-check codes incorporated with the features of GF(2^8) field in finite field, the present coding scheme could tolerate simultaneous failures of four blocks with only four redundancy blocks for a group of continuous data. Meanwhile, its encoding/decoding complexities had linear relationship with respect to the size of data blocks and it supported arbitrary size of data. Furthermore, it was proved to have MDS property, thus achieving optimal erasure-correcting capability with the same redundancy information. Theoretical and experimental analysis showed that the present coding scheme could code with 27 continuous data blocks at most.

Key words: erasure-correcting codes; bulk data; parity-check codes; GF(2^8) field; MDS

1 引言

随着 P2P 网络、多播传输等技术的发展，基于数据块或数据分组的大容量数据传输成为互联网应用的重要组成部分[1]。为解决网络反馈的拥塞问题，研究者提出了随机发送[2]、局部恢复[3]、分层恢复[4]等技术。这些方法能够有效提高数据传输的可靠性，但会导致显著的网络延迟，这在大容量数

收稿日期：2010-10-25；**修回日期**：2011-06-30
基金项目：上海市重点学科建设项目基金资助项目(B114)；AMD 大学合作计划基金资助项目
Foundation Items: Shanghai Leading Academic Discipline Project (B114); AMD University Cooperation Program

据的实时传输中是无法容忍的。研究表明，可以把基于前向纠错（FEC，forward error correcting）的纠删编码应用于大容量数据的可靠传输[5]。如果所采用的纠删编码的编解码运算复杂度低，还可以实现大容量数据的实时传输。现有的纠删编码一般采用面向比特或信息符号（symbol）的编码方法，且大多基于经典的 Reed-Solomon 码（简称 RS 码）[6]。这类编码的纠删能力较强，但编解码过程较为复杂。在处理以数据块或数据分组形式传输的大容量数据时，由于编解码运算复杂度较高，需要对大数据分组进行分割以形成若干短小的数据分组。在同样的差错率环境下，这些编码并不能有效减少数据重传的次数[7]。

长期以来，在数据块与数据块之间的纠删编码中，除了基于异或运算的校验和方法外，例如奇偶校验码（parity-check codes）[8]，很少有更强纠删能力的编码应用于以数据块或数据分组形式传输的海量信息系统中。为提高这类传输应用的可靠性，进一步增强编码的纠删能力，能够纠正 2 个及更多差错的编码方案成为信道编码领域的重要研究领域。近年来，不断涌现出一些基于阵列码（array codes）的纠删编码[9]。EVENODD 码[10]、X 码[11]、B 码[12]能够容忍单个或 2 个差错。然而，这些编码的纠删能力十分有限。能够容忍 3 个差错的编码中有 HOVER 码[13]和 WEAVER 码[14]，但这 2 种编码不是 MDS 码。根据编码理论[15]，上述 2 种编码并没有达到最佳的纠删能力。Feng 等[16,17]提出了 2 种基于 CPM（circular permutation matrices）矩阵的编码，分别能够容忍 3 个和 4 个差错。由于利用 CPM 矩阵的性质构造校验矩阵，使其编解码算法难以实现；解码过程采用线性方程高斯消元法求解，解码算法复杂度较高。除阵列码之外，基于 RS 码的纠删编码也获得了广泛研究。RS 码类的编码具有多纠删的特点，纠删能力较强，但它们的生成矩阵难以构造[18]。胡飞等[19]提出了一种基于 RS 码的数据分组层 FEC 编码的软件实现，将 FEC 编码进一步应用到分组交换网络，实现了数据分组之间的纠删。然而该编码不是 MDS 码，编码效率与纠删能力没有达到最优。万武南等[20]在 EVENODD 码的基础上，提出一种只需 3 个冗余数据块即可容忍任意 3 个数据块同时差错的 EEOD 码。虽然 EEOD 码具有 MDS 性质，然而其纠删能力仍然较为有限。Lacan 等[21]提出一种基于范德蒙矩阵的 MDS 纠删编码，采用 2 个范德蒙矩阵相乘的方法来构造任意子方阵都可逆的编码矩阵。该方法是构造基于范德蒙矩阵的 MDS 多纠删码编码矩阵的新方法，但是这种方法在编码和解码时都需要求矩阵的逆并计算矩阵乘积，编解码复杂度较高。AL-Shaikhi 等[22]针对数据分组丢失提出了基于范德蒙矩阵和移位操作的纠删编码，但是对编码的纠删能力缺乏理论证明，也没有体现移位操作对编码和解码算法复杂度的改善。

本文在奇偶校验码的基础上，利用有限域 GF(2^8)域的特性，提出一种新的面向大容量数据块的快速块间多纠删编码方案。该编码采用基于有限域运算的数据块间校验方法，只需生成 4 个冗余数据块即可实现对任意 4 个发生差错或丢失的数据块进行恢复，达到 MDS 码的理想纠删性能，相比现有同类纠删编码的纠删能力有较大提升。由于采用数据块间的校验，编码对数据块的大小没有限制。编解码运算的复杂度与数据块的大小成线性关系，编解码运算的开销较小，适用于面向大容量数据块传输的实时应用环境。

2　4 纠删编码方案

大容量数据的传输一般采用分块方式，处理对象为数据块或数据分组。为方便起见，本文把数据块或数据分组统称为数据块（大小以字节为单位）。假设数据块的大小为 l byte，用向量 $\boldsymbol{B}=(b_0,b_1,\cdots,b_{l-1})$ 表示。计算机中的数据大多以字节为单位，选择有限域 GF(2^8)作为编码的运算空间可以更高效地表示数据。如未做特殊说明，本文所有运算默认为在有限域 GF(2^8)上的运算。由于本文编码方案是在奇偶校验码的基础上扩展而来，因此在介绍具体的编码过程之前，先简要介绍一下奇偶校验码。

2.1　奇偶校验码

奇偶校验码利用 n 个源数据块生成一个大小相同的校验数据块。假设 $\boldsymbol{b}_0,\boldsymbol{b}_1,\cdots,\boldsymbol{b}_{n-1}$ 表示 n 个源数据块，$\boldsymbol{p}$ 表示生成的校验数据块，其编码过程可以表示为

$$\boldsymbol{b}_0 \oplus \boldsymbol{b}_1 \oplus \cdots \oplus \boldsymbol{b}_{n-1} = \boldsymbol{p} \tag{1}$$

其中，“⊕”表示异或运算（XOR），即有限域 GF(2^*)上的加法运算。当 n 个源数据块中有一个数据块 $\boldsymbol{b}_x\ (0 \leqslant x \leqslant n-1)$ 出现差错时，根据编码式（1）能够恢复差错数据块 $\boldsymbol{b}_x$：

$$\boldsymbol{b}_x = \boldsymbol{b}_0 \oplus \boldsymbol{b}_1 \oplus \cdots \oplus \boldsymbol{b}_{x-1} \oplus \boldsymbol{b}_{x+1} \oplus \cdots \oplus \boldsymbol{b}_{n-1} \oplus \boldsymbol{p} \tag{2}$$

即通过剩余 n 个正确数据块对应的数据进行异或运算得到。可见，奇偶校验码具有如下 2 个优势：①对源数据块的大小没有限制，适合面向大容量数据块的编码；②编码和解码运算只涉及到异或运算，算法的复杂度较低，运算速度快。奇偶校验码只能容忍单个差错，无法满足多纠删的应用需求。本文从编码的代数理论角度分析奇偶校验码，充分利用上述 2 个优势，结合有限域 GF(2^8)域的特性，在此基础上增加多纠删能力。

2.2 编码过程

对于 k 个大小相同的源数据块 $\boldsymbol{B}_0,\boldsymbol{B}_1,\cdots,\boldsymbol{B}_{k-1}$，采用纠删编码后能够容忍 1~4 个任意位置同时发生差错或丢失。编码方案仅增加 4 个冗余校验数据块 $\boldsymbol{C}_0,\boldsymbol{C}_1,\boldsymbol{C}_2,\boldsymbol{C}_3$，其生成过程也就是多纠删编码方案的编码过程。4 个冗余校验数据块由 4 个线性无关的校验方程生成，组成如下的校验方程组：

$$\begin{cases}\boldsymbol{B}_0\oplus\boldsymbol{B}_1\oplus\cdots\oplus\boldsymbol{B}_{k-1}=\boldsymbol{C}_0\\ \boldsymbol{B}_0\oplus\alpha^{1}\cdot\boldsymbol{B}_1\oplus\cdots\oplus\alpha^{k-1}\cdot\boldsymbol{B}_{k-1}=\boldsymbol{C}_1\\ \boldsymbol{B}_0\oplus\alpha^{2}\cdot\boldsymbol{B}_1\oplus\cdots\oplus\alpha^{2(k-1)}\cdot\boldsymbol{B}_{k-1}=\boldsymbol{C}_2\\ \boldsymbol{B}_0\oplus\alpha^{3}\cdot\boldsymbol{B}_1\oplus\cdots\oplus\alpha^{3(k-1)}\cdot\boldsymbol{B}_{k-1}=\boldsymbol{C}_3\end{cases}\tag{3}$$

其中，α是有限域 GF(2^8)的一个本原元。因此，k 个源数据块经过编码后得到 k+4 个数据块 $\boldsymbol{B}_0,\boldsymbol{B}_1,\cdots,\boldsymbol{B}_{k-1},\boldsymbol{C}_0,\boldsymbol{C}_1,\boldsymbol{C}_2,\boldsymbol{C}_3$。编码后的数据中包含源数据，由编码理论可知该编码属于系统编码，可以记为[k+4, k, 5]码。

2.3 纠删原理

编码方案[k+4, k, 5]能够容忍任意 4 个数据块同时差错。编码后的 k+4 个数据块中任意 4 个数据块（源数据块或冗余数据块）同时出现差错时，能够通过剩余的 k 个数据块加以恢复。换句话说，编码后的 k+4 个数据块中的任意 k 个数据块都能够重建 k 个源数据块。在本文的编码方案中，如果只是 4 个冗余数据块 $\boldsymbol{C}_0,\boldsymbol{C}_1,\boldsymbol{C}_2,\boldsymbol{C}_3$ 差错，采用校验方程组（3）即可重建。值得指出的是，在数据传输领域的实际应用中，一般只关注源数据的差错。当源数据能够正确传输时，不考虑冗余数据的差错情况，这样可以省去重建冗余数据的运算，从而加快译码时间。

源数据块出现差错，即 k 个源数据块中有 1 个、2 个、3 个或 4 个数据块同时发生差错。因为编码能够容忍任意 4 个数据块同时差错，所以必须存在与发生差错的源数据块相同数目的正确冗余数据块。如果 k 个源数据块 $\boldsymbol{B}_0,\boldsymbol{B}_1,\cdots,\boldsymbol{B}_{k-1}$ 中有 e 个数据块 $\boldsymbol{B}_{i_1},\boldsymbol{B}_{i_2},\cdots,\boldsymbol{B}_{i_e}$ 出错（1≤e≤4），则冗余数据块 $\boldsymbol{C}_0,\boldsymbol{C}_1,\boldsymbol{C}_2,\boldsymbol{C}_3$ 中必须存在 e 个正确的冗余数据块 $\boldsymbol{C}_{j_1},\boldsymbol{C}_{j_2},\cdots,\boldsymbol{C}_{j_e}$。为此，可以从校验方程组（3）中选取 e 个冗余数据块 $\boldsymbol{C}_{j_1},\boldsymbol{C}_{j_2},\cdots,\boldsymbol{C}_{j_e}$ 对应的方程形成新的方程组：

$$\begin{cases}\boldsymbol{B}_0\oplus\alpha^{j_1}\cdot\boldsymbol{B}_1\oplus\cdots\oplus\alpha^{j_1(k-1)}\cdot\boldsymbol{B}_{k-1}=\boldsymbol{C}_{j_1}\\ \boldsymbol{B}_0\oplus\alpha^{j_2}\cdot\boldsymbol{B}_1\oplus\cdots\oplus\alpha^{j_2(k-1)}\cdot\boldsymbol{B}_{k-1}=\boldsymbol{C}_{j_2}\\ \vdots\\ \boldsymbol{B}_0\oplus\alpha^{j_e}\cdot\boldsymbol{B}_1\oplus\cdots\oplus\alpha^{j_e(k-1)}\cdot\boldsymbol{B}_{k-1}=\boldsymbol{C}_{j_e}\end{cases}$$

把剩余（k–e）个正确的源数据块移至方程组右边，得到如下方程组：

$$\begin{cases}\alpha^{i_1j_1}\cdot\boldsymbol{B}_{i_1}\oplus\alpha^{i_2j_1}\cdot\boldsymbol{B}_{i_2}\oplus\cdots\oplus\alpha^{i_ej_1}\cdot\boldsymbol{B}_{i_e}=\boldsymbol{C}_{j_1}\oplus\left(\bigoplus_{\substack{t=0\\ t\neq i_1,\cdots,i_e}}^{k-1}\alpha^{j_1t}\cdot\boldsymbol{B}_t\right)\\ \alpha^{i_1j_2}\cdot\boldsymbol{B}_{i_1}\oplus\alpha^{i_2j_2}\cdot\boldsymbol{B}_{i_2}\oplus\cdots\oplus\alpha^{i_ej_2}\cdot\boldsymbol{B}_{i_e}=\boldsymbol{C}_{j_2}\oplus\left(\bigoplus_{\substack{t=0\\ t\neq i_1,\cdots,i_e}}^{k-1}\alpha^{j_2t}\cdot\boldsymbol{B}_t\right)\\ \cdots\\ \alpha^{i_1j_e}\cdot\boldsymbol{B}_{i_1}\oplus\alpha^{i_2j_e}\cdot\boldsymbol{B}_{i_2}\oplus\cdots\oplus\alpha^{i_ej_e}\cdot\boldsymbol{B}_{i_e}=\boldsymbol{C}_{je}\oplus\left(\bigoplus_{\substack{t=0\\ t\neq i_1,\cdots,i_e}}^{k-1}\alpha^{j_et}\cdot\boldsymbol{B}_t\right)\end{cases}\tag{4}$$

可以看出，方程组（4）为有限域 GF(2^8)上包含 e 个未知量 $\boldsymbol{B}_{i_1},\boldsymbol{B}_{i_2},\cdots,\boldsymbol{B}_{i_e}$ 的 e 个方程所组成的线性方程组。由克拉姆法则[23]可知，如果方程组的系数矩阵为非奇异矩阵（即可逆），则该方程组有唯一的解，发生差错的数据块 $\boldsymbol{B}_{i_1},\boldsymbol{B}_{i_2},\cdots,\boldsymbol{B}_{i_e}$ 可以恢复。

由方程组（4）的构造过程可知，其系数矩阵 $\begin{bmatrix}\alpha^{i_1j_1} & \alpha^{i_2j_1} & \cdots & \alpha^{i_ej_1}\\ \alpha^{i_1j_2} & \alpha^{i_2j_2} & \cdots & \alpha^{i_ej_2}\\ \vdots & \vdots & \ddots & \vdots\\ \alpha^{i_1j_e} & \alpha^{i_2j_e} & \cdots & \alpha^{i_ej_e}\end{bmatrix}$ 为 e×e 阶方阵，即校验方程组（3）的 4×k 阶系数矩阵 $\boldsymbol{V}=\begin{bmatrix}1 & 1 & \cdots & 1\\ 1 & \alpha^{1} & \cdots & \alpha^{k-1}\\ 1 & \alpha^{2} & \cdots & \alpha^{2(k-1)}\\ 1 & \alpha^{3} & \cdots & \alpha^{3(k-1)}\end{bmatrix}$ 的任意 e×e 阶子方阵。因此，只要校验方程组（3）

的 4×k 阶系数矩阵 $\boldsymbol{V}$ 在有限域 GF(2^8)上的任意 e×e 阶的子方阵都是非奇异的，则方程组（4）必有唯一解，使得编码能够容忍 4 个数据块同时差错。本文给出一个定理以表明：在选择合适的生成多项式（即本原多项式）构造的有限域 GF(2^8)上，适当选择本原元α后，可以构造出具备上述性质的校验方程组的系数矩阵 $\boldsymbol{V}$，并且可以求得能够编码的数据块个数 k 的最大取值。

定理 1　在有限域 GF(2^8)上，存在 4×k 阶范德蒙矩阵（α是有限域 GF(2^8)上的一个本原元）

$$\boldsymbol{V}=\begin{bmatrix}1 & 1 & \cdots & 1\\ 1 & \alpha^{1} & \cdots & \alpha^{k-1}\\ 1 & \alpha^{2} & \cdots & \alpha^{2(k-1)}\\ 1 & \alpha^{3} & \cdots & \alpha^{3(k-1)}\end{bmatrix}$$

其任意 e×e 阶子方阵（e = 1,2,3,4）都是非奇异矩阵；选择适当的本原元α，矩阵 $\boldsymbol{V}$ 的列数 k 能够取得最大值 27。

证明　对于 4×k 阶的矩阵 $\boldsymbol{V}$，其任意阶子方阵可以分成 4 类，分别对应于 e=1、2、3、4 时的 1×1 阶、2×2 阶、3×3 阶与 4×4 阶子方阵。下面按照复杂度依次证明这 4 类任意子方阵都是非奇异矩阵。

1) 1×1 阶子方阵

矩阵 $\boldsymbol{V}$ 的任意 1×1 阶子方阵为矩阵的任意元素，可以表示为 $\boldsymbol{M}_1=\left[\alpha^{i}\right]$（0≤$i$≤$k$−1）。由于$\alpha$是有限域 GF($2^8$)的本原元，所以对任意的 i 都有 $\alpha^{i}\neq 0$，即行列式 $\left|\boldsymbol{M}_1\right|=\alpha^{i}\neq 0$。因此，矩阵 $\boldsymbol{V}$ 的任意 1×1 阶子方阵都是非奇异的。

2) 4×4 阶子方阵

矩阵 $\boldsymbol{V}$ 的任意 4×4 阶子方阵由其任意 4 列构成，可以表示为

$$\boldsymbol{M}_4=\begin{bmatrix}1 & 1 & 1 & 1\\ \alpha^{p} & \alpha^{q} & \alpha^{r} & \alpha^{s}\\ \alpha^{2p} & \alpha^{2q} & \alpha^{2r} & \alpha^{2s}\\ \alpha^{3p} & \alpha^{3q} & \alpha^{3r} & \alpha^{3s}\end{bmatrix}$$

其中，p、q、r、s（0≤p<q<r<s≤k−1）表示矩阵中任意不同的 4 列。显然，$\boldsymbol{M}_4$ 是 4×4 阶的范德蒙矩阵。根据有限域理论，由于α为本原元，故对任意的 i 和 j，当 0≤i≠j≤255 时都有 $\alpha^{i}\neq\alpha^{j}$。所以当 k≤255 时，范德蒙矩阵 $\boldsymbol{M}_4$ 中的元素 $\alpha^{p},\alpha^{q},\alpha^{r},\alpha^{s}$ 互不相等。根据范德蒙矩阵的性质，$\boldsymbol{M}_4$ 的行列式不为 0。因此，矩阵 $\boldsymbol{V}$ 的任意 4×4 阶子方阵都是非奇异的。

3) 2×2 阶子方阵

矩阵 $\boldsymbol{V}$ 的任意 2×2 阶子方阵由其任意 2 行 2 列构成，可以表示为

$$\boldsymbol{M}_2=\begin{bmatrix}\alpha^{pi} & \alpha^{qi}\\ \alpha^{pj} & \alpha^{qj}\end{bmatrix}$$

其中，i 和 j 表示矩阵 $\boldsymbol{V}$ 的任意 2 行（0≤i<j≤3）；p 和 q 表示矩阵 V 的任意 2 列（0≤p<q≤k−1）。矩阵 $\boldsymbol{M}_2$ 的行列式为

$$\left|\boldsymbol{M}_2\right|=\alpha^{pi}\cdot\alpha^{qj}\oplus\alpha^{pj}\cdot\alpha^{qi}=\alpha^{pi+qj}\oplus\alpha^{pj+qi}$$

证明矩阵 $\boldsymbol{M}_2$ 为非奇异矩阵，即证明 $\alpha^{pi+qj}\oplus\alpha^{pj+qi}\neq 0$。在有限域上，$\alpha^{pi+qj}\oplus\alpha^{pj+qi}\neq 0$ 相当于 $\alpha^{pi+qj}\neq\alpha^{pj+qi}$。在 GF($2^8$)域上，上述不等式可以进一步转化为$(q-p)(j-i)\neq 255t$（$t$=0,1,2,…）。因为 0≤$i$<$j$≤3 且 0≤$p$<$q$≤$k$−1，所以 0<($j$−$i$)≤3 且 0<($q$−$p$) ≤($k$−1)，故有 3($k$−1)≠255t。因此，当 t=1 时，满足 $\left|\boldsymbol{M}_2\right|\neq 0$ 的 k 取到最大值 $\left\lfloor\frac{255}{3}\right\rfloor+1=86$。即当 k<86 时，矩阵 $\boldsymbol{V}$ 的任意 2×2 阶子方阵都是非奇异的。

4) 3×3 阶子方阵

矩阵 $\boldsymbol{V}$ 的任意 3×3 阶子方阵由其任意 3 行 3 列构成。由于矩阵 $\boldsymbol{V}$ 的行数是固定值 4，因此可以按行的取法把 3×3 阶子方阵分为 4 类（$C_4^{3}=4$）分别加以论证。不妨设 p、q、r 表示所选取的 3 列下标，则有 0≤p<q<r≤k−1。按照所取的行号（123，234，124，134），可以表示成如下 4 种形式：

$$\boldsymbol{M}_{123}=\begin{bmatrix}1 & 1 & 1\\ \alpha^{p} & \alpha^{q} & \alpha^{r}\\ \alpha^{2p} & \alpha^{2q} & \alpha^{2r}\end{bmatrix}$$

$$\boldsymbol{M}_{234}=\begin{bmatrix}\alpha^{p} & \alpha^{r} & \alpha^{s}\\ \alpha^{2p} & \alpha^{2r} & \alpha^{2s}\\ \alpha^{3p} & \alpha^{3r} & \alpha^{3s}\end{bmatrix}$$

$$\boldsymbol{M}_{124}=\begin{bmatrix}1 & 1 & 1\\ \alpha^{p} & \alpha^{q} & \alpha^{r}\\ \alpha^{3p} & \alpha^{3q} & \alpha^{3r}\end{bmatrix}$$

$$\boldsymbol{M}_{134}=\begin{bmatrix}1 & 1 & 1\\ \alpha^{2p} & \alpha^{2q} & \alpha^{2r}\\ \alpha^{3p} & \alpha^{3q} & \alpha^{3r}\end{bmatrix}$$

可以看出，子方阵 $\boldsymbol{M}_{123}$ 为范德蒙矩阵，而子方阵 $\boldsymbol{M}_{234}$ 可以通过简单的变换（对各列提取公因子 $\alpha^p,\alpha^q,\alpha^r$）变为范德蒙矩阵。因此，这 2 种形式的任意 3×3 阶子方阵都是非奇异矩阵。子方阵 $\boldsymbol{M}_{124}$ 与 $\boldsymbol{M}_{134}$ 为缺行的范德蒙矩阵，其行列式计算方法如下[24]：

$$\left|\boldsymbol{M}_{124}\right|=(\alpha^p\oplus\alpha^q\oplus\alpha^r)\prod_{\substack{i<j\\ i,j=p,q,r}}(\alpha^i\oplus\alpha^j)$$

$$\left|\boldsymbol{M}_{134}\right|=(\alpha^{p+q}\oplus\alpha^{p+r}\oplus\alpha^{q+r})\prod_{\substack{i<j\\ i,j=p,q,r}}(\alpha^i\oplus\alpha^j)$$

要证明子方阵 $\boldsymbol{M}_{124}$ 与 $\boldsymbol{M}_{134}$ 非奇异，即证明对任意的 p、q 和 r（$0\leqslant p<q<r\leqslant k-1$），都有 $\alpha^p\oplus\alpha^q\oplus\alpha^r\neq 0$ 且 $\alpha^{p+q}\oplus\alpha^{p+r}\oplus\alpha^{q+r}\neq 0$。基于 GF($2^8$)域的运算，可以转化为证明存在 k，对任意 p 和 q（$0\leqslant p<q\leqslant k-1$），都有不等式 $\alpha^{q-p}\oplus\alpha^{r-p}\neq 1$ 且 $\alpha^{r-q}\oplus\alpha^{r-p}\neq 1$ 成立。在有限域 GF(2^8)上，可以进一步转化为求解满足等式

$$\alpha^p\oplus\alpha^q=1 \tag{5}$$

的 k 的最小值，即满足式（5）的 q 值的上限。由于 GF(2^8)域有多个不同的本原元，所以需要考虑不同的本原元α下能够满足式（5）的 q 的最小值。根据有限域理论，有限域的本原元是本原多项式的根。GF(2^8)域共有 16 个不同的本原多项式，并且同一个本原多项式有 8 个共轭根 $\alpha,\alpha^2,\alpha^{2^2},\cdots,\alpha^{2^7}$。同时，在有限域 GF($2^8$)上有性质：$(\alpha^2)^p\oplus(\alpha^2)^q=(\alpha^p\oplus\alpha^q)^2$。因此，同一个本原多项式对应的所有本原元满足式（5）的 p 和 q 值相同。因此，只需要考虑不同的本原多项式对应的一个本原元（不妨取最小的本原元）。这样，在有限域 GF(2^8)上只需测试 16 个不同的本原元对应的满足式（5）的 q 的最小值。GF(2^8)域上不同的本原元α对应的满足等式 $\alpha^p\oplus\alpha^q=1$ 的 q 的最小值如表 1 所示。从表 1 的测试结果能够看出，对于 GF(2^8)域的不同本原元（即本原多项式），满足式（5）的 q 的最小值不完全相同。选择适当的本原多项式所对应的本原元，可以得到满足式（5）的 k 的最大值：即 k=27 时，方阵 $\boldsymbol{M}_{124}$ 与 $\boldsymbol{M}_{134}$ 都是非奇异的。因此，在选择适当的本原元后，当 $k\leqslant 27$，矩阵 $\boldsymbol{V}$ 的任意 3×3 阶子方阵都是非奇异矩阵。

表 1　GF(2^8)域上不同的本原元α对应的满足等式 $\alpha^p\oplus\alpha^q=1$ 的 q 的最小值

本原多项式	p	q
$x^8+x^4+x^3+x^2+1$	10	21
$x^8+x^5+x^3+x+1$	12	13
$x^8+x^5+x^3+x^2+1$	15	16
$x^8+x^6+x^3+x^2+1$	1	23
$x^8+x^6+x^4+x^3+x^2+x+1$	11	13
$x^8+x^6+x^5+x+1$	9	20
$x^8+x^6+x^5+x^2+1$	22	23
$x^8+x^6+x^5+x^3+1$	1	16
$x^8+x^6+x^5+x^4+1$	11	21
$x^8+x^7+x^2+x+1$	8	27
$x^8+x^7+x^3+x^2+1$	11	20
$x^8+x^7+x^5+x^3+1$	1	13
$x^8+x^7+x^6+x+1$	19	27
$x^8+x^7+x^6+x^3+x^2+x+1$	2	27
$x^8+x^7+x^6+x^5+x^2+x+1$	25	27
$x^8+x^7+x^6+x^5+x^4+x^2+1$	2	13

综上，有限域 GF(2^8)上存在 4×k 阶矩阵 $\boldsymbol{V}$，其任意阶子方阵都是非奇异的；选择适当的本原元，k 可以取到最大值 27。

证毕

由上述定理可知，通过矩阵 $\boldsymbol{V}$ 构造的校验方程组（3）能够恢复任意 4 个源数据块的差错，其最小 Hamming 距离为 5。根据编码理论，该编码属于 MDS 码。

2.4　解码及优化

纠删编码的解码是当源数据块出现差错时，利用剩余的正确数据块，通过解码算法重建差错数据块。由于冗余数据块的作用主要是用来辅助源数据块可靠传输，因此一般不考虑冗余数据块出错的情况，即解码算法只考虑一个或多个源数据块出错的情况。由于重建源数据块的解码方法非常类似，不失一般性，本文只给出最复杂情况的解码过程，即 4 个源数据块出错的情况。

假设对 k 个源数据块 $\boldsymbol{B}_0,\boldsymbol{B}_1,\cdots,\boldsymbol{B}_{k-1}$ 经过纠删编码后的 k+4 个数据块为 $\boldsymbol{B}_0,\boldsymbol{B}_1,\cdots,\boldsymbol{B}_{k-1},\boldsymbol{C}_0,\boldsymbol{C}_1,\boldsymbol{C}_2,\boldsymbol{C}_3$，通过信道传输至信宿后，有 4 个源数据块 $\boldsymbol{B}_p,\boldsymbol{B}_q,\boldsymbol{B}_r,\boldsymbol{B}_s$ 出错($0\leqslant p<q<r<s\leqslant k-1$)。根据 2.3 节的方程组(4)，得到解码方程组：

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ \alpha^{p} & \alpha^{q} & \alpha^{r} & \alpha^{s} \\ \alpha^{2p} & \alpha^{2q} & \alpha^{2r} & \alpha^{2s} \\ \alpha^{3p} & \alpha^{3q} & \alpha^{3r} & \alpha^{3s} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{B}_p \\ \boldsymbol{B}_q \\ \boldsymbol{B}_r \\ \boldsymbol{B}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{P}_0 \\ \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \\ \boldsymbol{P}_3 \end{bmatrix} \tag{6}$$

其中，

$$\boldsymbol{P}_i = \boldsymbol{C}_i \oplus \left(\bigoplus_{\substack{t=0 \\ t\neq p,q,r,s}}^{k-1} \alpha^{i} \cdot \boldsymbol{B}_t \right), \quad i = 0,1,2,3 \tag{7}$$

2.3 节中已经证明了方程组（6）的系数矩阵 $\boldsymbol{M}$ 可逆，由 Cramer 法则可知方程组的解为

$$\begin{bmatrix} \boldsymbol{B}_p \\ \boldsymbol{B}_q \\ \boldsymbol{B}_r \\ \boldsymbol{B}_s \end{bmatrix} = \left(\frac{adj\boldsymbol{M}}{|\boldsymbol{M}|} \right) \cdot \begin{bmatrix} \boldsymbol{P}_0 \\ \boldsymbol{P}_1 \\ \boldsymbol{P}_2 \\ \boldsymbol{P}_3 \end{bmatrix} \tag{8}$$

其中，$adj\boldsymbol{M}$ 为矩阵 $\boldsymbol{M}$ 的伴随矩阵，记为 $adj\boldsymbol{M} = (A_{ij})$（$0 \leqslant i,j \leqslant 3$）；$A_{ij}$ 为系数矩阵 $\boldsymbol{M}$ 的代数余子式（3×3 的行列式），且 $|\boldsymbol{M}| = \bigoplus_{j=0}^{3} A_{0j}$。

从上述解码方程式可以看出，解码过程分为 3 步。

步骤 1　根据差错数据块的位置，按照方程（7）计算 $\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{P}_3$。

步骤 2　根据差错数据块的位置，计算矩阵 $\left(\frac{adj\boldsymbol{M}}{|\boldsymbol{M}|} \right)$。不难看出，这部分运算的复杂度与数据块大小无关，只与差错数目相关。当差错数据块的个数确定后，该步骤的运算量固定。

步骤 3　按照方程（8）重建差错的数据块。

在解码过程中，步骤 1 与步骤 3 只涉及乘加运算，运算复杂度较低。步骤 2 的运算复杂度稍高，需要计算 4×4 阶矩阵 $\boldsymbol{M}$ 的逆矩阵。然而，这一步的运算仅依赖于差错数据块的个数，而与源数据块的总量及数据块本身的大小无关。因此，当差错数目及差错位置确定之后，该步骤的运算可以先于其他 2 步完成。由于运算量不依赖于数据块本身的大小，因此适用于大容量数据块的应用环境。

为了降低解码运算的开销，使其达到实时的应用需求，可以从 2 个方面对解码算法中的步骤 2 进行优化。首先，可以对步骤 2 中 4×4 阶矩阵 $\boldsymbol{M}$ 进行求逆运算的分解。根据代数理论，矩阵 $\boldsymbol{M}$ 的逆 $\boldsymbol{M}^{-1} = \frac{adj\boldsymbol{M}}{|\boldsymbol{M}|}$。$adj\boldsymbol{M}$ 可以表示为

$$adj\boldsymbol{M} = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix}$$

其中，a_{ij}（$i,j = 1,2,3,4$)为矩阵 $\boldsymbol{M}$ 的代数余子式（3×3 阶行列式），由矩阵 $\boldsymbol{M}$ 可得

$$a_{11} = \begin{vmatrix} \alpha^{q} & \alpha^{r} & \alpha^{s} \\ \alpha^{2q} & \alpha^{2r} & \alpha^{2s} \\ \alpha^{3q} & \alpha^{3r} & \alpha^{3s} \end{vmatrix}, \cdots, a_{14} = \begin{vmatrix} \alpha^{p} & \alpha^{q} & \alpha^{r} \\ \alpha^{2p} & \alpha^{2q} & \alpha^{2r} \\ \alpha^{3p} & \alpha^{3q} & \alpha^{3r} \end{vmatrix}$$

$$\cdots,$$

$$a_{41} = \begin{vmatrix} 1 & 1 & 1 \\ \alpha^{q} & \alpha^{r} & \alpha^{s} \\ \alpha^{2q} & \alpha^{2r} & \alpha^{2s} \end{vmatrix}, \cdots, a_{14} = \begin{vmatrix} 1 & 1 & 1 \\ \alpha^{p} & \alpha^{q} & \alpha^{r} \\ \alpha^{2p} & \alpha^{2q} & \alpha^{2r} \end{vmatrix}$$

由于矩阵 $\boldsymbol{M}$ 的第 1 行都是 1，所以有 $|\boldsymbol{M}| = a_{11} \oplus a_{12} \oplus a_{13} \oplus a_{14}$。因此，只需计算矩阵 $\boldsymbol{M}$ 的 16 个代数余子式 $a_{11}, a_{12}, \cdots, a_{44}$ 即可计算出 $\frac{adj\boldsymbol{M}}{|\boldsymbol{M}|}$。由线性代数中行列式理论可知，3×3 阶行列式 $\begin{vmatrix} \alpha^{ip} & \alpha^{iq} & \alpha^{ir} \\ \alpha^{jp} & \alpha^{jq} & \alpha^{jr} \\ \alpha^{kp} & \alpha^{kq} & \alpha^{kr} \end{vmatrix}$ 可以表示为

$$\alpha^{ip}\alpha^{jq}\alpha^{kr} - \alpha^{ip}\alpha^{jr}\alpha^{kq} - \alpha^{iq}\alpha^{jp}\alpha^{kr} + \alpha^{iq}\alpha^{jr}\alpha^{kp} + \alpha^{ir}\alpha^{jp}\alpha^{kq} + \alpha^{ir}\alpha^{jq}\alpha^{kp}$$

在有限域 GF(2^8)上，又可以表示为

$$\alpha^{ip}\alpha^{jq}\alpha^{kr} \oplus \alpha^{ip}\alpha^{jr}\alpha^{kq} \oplus \alpha^{iq}\alpha^{jp}\alpha^{kr} \oplus \alpha^{iq}\alpha^{jr}\alpha^{kp} \oplus \alpha^{ir}\alpha^{jp}\alpha^{kq} \oplus \alpha^{ir}\alpha^{jq}\alpha^{kp}$$

有限域上的运算采用基于模的运算，有限域中的元素用二进制表示后，加法运算对应计算机中的异或运算，并且有限域上的加法和减法相同。对于有限域上的乘法和除法，则为模一个不可约多项的多项式乘、除运算。如果按照定义现实乘法和除法，运算相对比较复杂。为此，一些计算机硬件实现了有限域上的乘除法运算部件，大大加快了有限域上的运算速度。然而这类专有硬件通用性较差，制造成本较高，在一般的计算机系统中难以实现。因此，可以采用软件的方法优化有限域上的乘除运算。由于有限域元素个数有限，并且域上的运算都是封闭的，因此一个确定的有限域中的所有元素之间的运

算结果可以枚举（如 GF(2^8)域）。基于此，可以利用查表的方法来优化有限域上的运算。由于有限域的所有元素都可以表示成一个本原元的幂，因此对于有限域 GF(2^8)上的任意 2 个元素 x, y 及一个本原元α，有：

$$x \otimes y = \alpha^i \alpha^j = \alpha^{i+j}$$

其中，$i = \log_\alpha^x, j = \log_\alpha^y$。由于本原元 $\alpha^{2^8-1} = \alpha^2 55 = 1$，令 $Q = 255$，则有 $\alpha^Q = 1$。因此，可以得到：

$$x \otimes y = \alpha^{(i+j)\%Q}$$

这样，只要事先构造出本原元α的对数表 unsigned char $t_{\log}$[256] 与指数表 unsigned char $t_{\exp}$[256]，有限域 GF(2^8)上的乘法运算可以采用图 1 所示的查表法进行优化。

```
# define Q 255
unsigned char gmul(const unsigned char x ,const unsigned char y)
{
    if (x == 0 || y == 0) return 0;
    return t_exp[(t_log[x] + t_log[y])%Q];
}
```

图 1 C 程序实现有限域 GF(2^8)上乘法查表法的代码

相比按运算定义实现的乘法，查表法在很大程度上降低了运算的时间复杂度，但是增加可运算的空间复杂度，即占用了 2Qbyte 的额外存储空间。从图 1 中的代码可以看出，算法需要进行 2 次查表，在查完对数表之后，仍然需要进行模的运算，而这一运算在计算机中的执行代价较高。为了解决这一问题，可以利用以空间换时间的方法来扩大指数表与对数表。这种方式虽然又占用了一部分额外的存储空间（指数表占用了 4Q+1byte 的存储空间，对数表仍然占用 Q byte 的存储空间），但是能够消除模运算，进一步优化了乘除法运算。定义指数表和对数表的方法如下[25]：

$$t_{\exp}[i] = \begin{cases} x,\ i \in [0, Q-1], x = \alpha^i \\ t_{\exp}[i\%Q],\ i \in [Q, 2Q-1] \\ 0,\ i \in [2Q, 4Q] \end{cases}$$

$$t_{\log}[i] = \begin{cases} 2Q, x = 0 \\ i, x \in [1, Q], x = \alpha^i \end{cases}$$

因此，图 1 中的算法可以按照上述公式予以化简，如图 2 所示。

```
#define Q 255
unsigned char gmul(const unsigned char x, const unsigned char y)
{
    return t_exp[(t_log[x] + t_log[y])];
}
unsigned char gdiv(const unsigned char x, const unsigned char y)
{
    return t_exp[(t_log[x] + Q - t_log[y])];
}
```

图 2 化简后的有限域 GF(2^8)上乘法查表法的代码

采用查表法对有限域上的乘除法进行优化后，3×3 阶代数余子式求值的代码如图 3 所示（其中，i、j、k 为行下标，p、q、r 为列下标）。

```
#define det 3×3(i,j,k,p,q,r) do {\
    t_exp[i*p+j*q+k*r] \
    ^t_exp[i*p+j*r+k*q] \
    ^t_exp[i*q+j*p+k*r] \
    ^t_exp[i*q+j*r+k*p] \
    ^t_exp[i*r+j*p+k*q] \
    ^t_exp[i*r+j*q+k*p] \
    } while (0)
```

图 3 C 程序实现 3×3 阶代数余子式求值的代码

在解码过程的 3 个步骤中，大量涉及到有限域 GF(2^8)上形如 $\alpha^x \cdot \boldsymbol{b}_i$ 的乘法运算。为了加速计算，可以选择本原元 x（在计算机里表示为 0x02，故乘法相当于移位操作）。因此，运算 $\alpha^x \cdot \boldsymbol{b}_i$ 相当于对数据块 $\boldsymbol{b}_i$ 中每个字节的二进制数据左移 x 位。可见，如果计算机硬件支持大规模数据移位运算，则解码算法的复杂度会大幅降低。对于没有相关硬件支持的平台，可以采用上面介绍的有限域乘除法查表法实施优化，从而利用空间复杂度的增加换取时间复杂度的减小。

3 性能分析

本文从编码方案的编码效率（或存储效率）与编解码算法的复杂度这 2 方面来讨论编码方案的性能。由 2.3 节的定理可知，编码方案[k+4, k, 5]属于 MDS 码，编码效率为 $\frac{k}{k+4}$。根据编码理论，编码效率已经达到 4 纠删码的最佳存储效率。

编码方案的编码过程实际上是通过编码校验方程组（3）产生 4 个冗余数据块的过程。因此，由校验方程组（3）可知，编码算法的复杂度与源数据块个数 k 及源数据块大小有关。编码算法的基本运算为 $\alpha^x \cdot b$（b 为大小为 l 个字节的源数据块）。

当本原元α选择多项式 x（表示为 0x02）时，编码的过程实际上是对长度为 l 的字节向量 b 做整体左移 x 位的运算。如果计算机硬件支持数据块的移位操作，那么运算 $\alpha^x \cdot b$ 只需要一步运算。因此，单个源数据块的编码运算复杂度为 $O(1)$，k 个源数据块的编码运算复杂度为 $O(k)$。当源数据块的个数 k 确定时，即编码长度固定时，编码的运算复杂度为 $O(1)$，不随着数据块的大小而变化。因此，较低的运算复杂度使其适用于大容量数据块的实际应用中。如果计算机硬件不支持数据块移位操作，可以利用查表法进行加速，即 $\alpha^x \cdot b = t_{\exp}[x + t_{\log}[b]]$。可见，对大小为 l byte 的单个源数据块编码的运算复杂度为 $O(l)$，对 k 个源数据块编码的运算复杂度为 $O(kl)$。因此，当编码的长度固定时，编码运算的复杂度为 $O(l)$，与数据块的大小 l 成线性关系。

由 2.4 节的解码过程可知，编码方案的解码分为 3 个步骤，每个步骤的运算量大小依赖于发生差错的源数据块的个数。不难看出，当只有一个源数据块出错时，解码过程即为奇偶校验码。不失一般性，对最复杂的情形（即 4 个源数据块同时出错）下的解码运算复杂度给予分析。在解码运算的 3 个步骤中，步骤 1 的运算过程与编码过程类似。因此，该部分的运算复杂度与编码过程的复杂度相同，对于大小为 l 个字节的数据块，其运算复杂度为 $O(l)$。步骤 2 实际上是计算 16 个 3×3 阶行列式，很显然，这部分运算的复杂度与数据块的大小无关，只依赖于发生差错的数据块的个数。当差错数据块的个数及差错位置确定时，步骤 2 的运算量恒定。因此，其算法复杂度为 $O(1)$。由于步骤 2 是整个解码算法的核心，且该部分的运算量恒定，与数据块的大小无关，再次表明编码方案的解码过程适用于面向大数据块的实际应用领域。步骤 3 是根据式（8）恢复出 4 个发生差错的数据块，对于数据块中的每一个字节来说，这部分的运算只是简单的“4 乘 3 加”运算。因此，其运算复杂度为 $O(4\times(4+3)\times l)=O(l)$。通过对解码过程的 3 个步骤的分析可知，整个解码算法的复杂度为 $O(4\times l) + O(1) + O(1) = O(l)$。可以看出，解码算法的运算复杂度较低。

当编码方案的编码长度（即源数据块的个数 k）及发生差错的数据块个数和差错位置确定之后，解码运算即可分成 2 个部分：一部分是运算量与数据块大小无关的步骤 2，该部分可以在重建数据块的每个字节前预先计算出来，计算结果可以重复应用于数据块中的每一个字节，大幅减少了解码运算量；另一部分是与数据块大小相关的字节加权运算，即有限域上的乘加运算，其运算复杂为 $O(l)$。图 4 比较了源数据块个数 k=8 的情况下，发生差错的数据块个数分别为 e=1，2，3，4 时，解码运算量与数据块大小的关系。图 5 比较了不同的编码长度（即源数据块个数 k=4，8，16，27）下有 4 个数据块发生差错时，解码运算量与数据块大小的关系。可以看出，当编码长度确定时，解码运算量与数据块的大小成线性关系，进而验证了本文编码方案的解码运算复杂度为 $O(l)$。

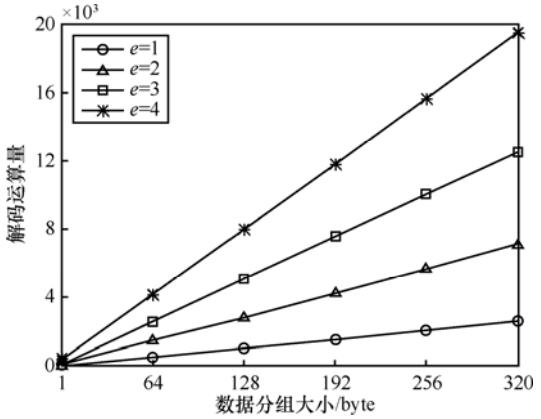


图 4　解码运算量与数据分组大小的对比（源数据块数目 k=8）

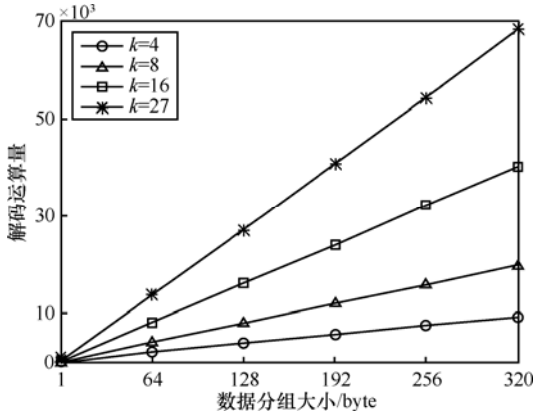


图 5　解码运算量与数据分组大小的对比（4 个源数据块出错）

为了便于全面了解本文编码的特点，把本文编码方案与 EVENODD 码、VCR 码[26]做纵向比较，主要包括以下 3 个指标：编码效率、能容忍出错的数据块个数和复杂度。对于 n 个源数据块的传输，比较结果如表 2 所示。

表 2　　3 种纠删码的比较

编码方案	编码效率	容忍数据块个数	复杂度
EVENODD	$k/(k+2)$	2	$O(n^2)$

VCR	$k/(k+3)$	3	$O(n^2)$
本文编码	$k/(k+4)$	4	$O(n)$

从表 2 可以看出，虽然本文编码方案的编码效率略低于 EVENODD 码和 VCR 码，但纠删能力高于它们。此外，利用查找表方式进行优化仅仅牺牲了很小一部分存储空间用于保存指数表和对数表，获得了线性时间复杂度的编解码性能。

4 结束语

针对面向大容量数据块的实际应用领域的特点，本文在奇偶校验码的基础上结合有限域 GF(2^8)的特性，提出了一种面向大数据块的快速多纠删编码方案。该编码方案最多能够容许任意 4 个数据块同时差错或丢失，编解码算法对数据块的大小没有限制且运算复杂度较低。同时，该编码属于 MDS 码，在同等冗余条件下具备最佳的纠删能力。最后，该编码方案最大能够对连续 27 个数据块进行编码。本文提出的 4 纠删编码方案不但在大容量数据实时传输中有着较好的应用价值，而且在分布式存储系统和 RAID 系统容错等领域也有着广阔的应用前景。下一步的工作是研究有限域 GF(2^8)域及其扩域上的编码方法，通过增加校验方程进一步提高编码方案的纠删能力，实现能够容忍 5 个甚至更多差错的纠删编码。

参考文献：

[1] COSTELLO D J, HAGENAUER J, IMAI H, *et al*. Applications of error-control coding[J]. IEEE Transactions on Information Theory, 1998, 44(6): 2531-2560.

[2] FLOYD S, JACOBSON V, MCCANNE S, *et al*. A reliable multicast framework for light-weight sessions and application level framing[J]. IEEE/ACM Transactions on Networking, 1997, 5(6): 784-803.

[3] LIN J C, PAUL S. RMTP: A Reliable multicast transport protocol[A]. IEEE INFOCOM1996[C]. San Francisco, CA, USA, 1996. 1414-1424.

[4] MCCANNE S, JACOBSON V, VETTERLI M. Receiver driven layered multicast[A]. IEEE INFOCOM1996[C]. San Francisco, CA, USA, 1996. 117-130.

[5] RIZZO L. Effective erasure codes for reliable computer communication protocols[J]. ACM Computer Communication Review, 1997, 27(2): 24-36.

[6] REED I, SOLOMON G. Polynomial codes over certain finite fields[J]. Journal of the SIAM, 1960, 8(2): 300-304.

[7] BYERS J W, LUBY M, MITZENMACHER M, *et al*. A digital fountain approach to reliable distribution of bulk data[J]. IEEE Journal on Selected Areas in Communications, 2002, 20(8):1528-1540.

[8] PATTERSON D A, GIBSON G A, KATZ R H. A case for redundant arrays of inexpensive disks (RAID)[A]. ACM SIGMOD1988[C]. New York, NY, USA, 1988. 109-116.

[9] PLANK J S, XU L. Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications[A]. IEEE NCA2006[C]. Cambridge, MA, 2006. 173-180.

[10] BLAUM M, BRADY J, BRUCK J, *et al*. EVENODD: an efficient scheme for tolerating double disk failures in raid architectures[J]. IEEE Transactions on Computers, 1995, 44(2): 192-202.

[11] XU L, BRUCK J. X-Code: MDS array codes with optimal encoding[J]. IEEE Transactions on Information Theory, 1999, 45(1): 272–275.

[12] XU L, BOHOSSIAN V, BRUCK J, *et al*. Low density MDS codes and factors of complete graphs[J]. IEEE Transactions on Information Theory, 1999, 45(6): 1817-1826.

[13] HAFNER J L. HoVer Erasure Codes for Disk Arrays[R]. IBM Research Division, FJ10352(A0507-015), 2005.

[14] HAFNER J L. WEAVER codes: Highly fault tolerant erasure codes for storage systems[A]. ACM FAST2005[C]. Berkeley, CA, USA, 2005. 211-224.

[15] MACWILLIAMS F J, SLOANE N J A. The Theory of Error Correcting Codes[M]. North-holland Publishing Company, 1977.

[16] FENG G L, DENG R, BAO F, *et al*. New efficient MDS array codes for RAID, part I: reed Solomon like codes for tolerating three disk failures[J]. IEEE Transactions on Computers, 2005, 54(9): 1071-1080.

[17] FENG G L, DENG R, BAO F, *et al*. New efficient MDS array codes for RAID, part II: rabin like codes for tolerating multiple (4) disk failures[J]. IEEE Transactions on Computers, 2005, 54(12): 1473-1482.

[18] MORELOS Z, ROBERT H. The Art of Error Correcting Coding[M]. John Wiley & Sons Inc, 2006.

[19] 胡飞, 朱耀庭, 朱光喜. 基于 Galois 域 Reed-Sololmon 码的数据分组层 FEC 编码软件实现[J]. 通信学报, 2003, 23(3): 57-64.
HU F, ZHU Y T, ZHU G X. A software implementation of packet-level FEC coding based on Reed-Solomon code over Galois field[J]. Journal

on Communications, 2003, 23(3): 57-64.

[20] 万武南, 吴震, 陈运等. 一种基于 3 容错阵列码的 RAID 数据布局[J]. 计算机学报, 2007, 30(10): 1721-1730.
WAN W N, WU Z, CHEN Y, *et al*. A data placement based on toleration on triple failures array codes in RAID[J]. Chinese Journal of Computers, 2007, 30(10): 1721-1730.

[21] LACAN J, FIMES J. Systematic MDS erasure codes based on Vandermonde matrices[J]. IEEE Communications Letters, 2004, 8(9): 570-572.

[22] AL-SHAIKHI A A, ILOW J. Packet loss recovery codes based on Vandermonde matrices and shift operators[A]. IEEE ISIT2008[C]. Toronto, ON, Canada, 2008. 1058-1062.

[23] 万哲先. 代数和编码(第三版)[M]. 北京：高等教育出版社, 2007.
WAN Z X. Algebra and Coding[M]. Beijing: Higher Education Press, 2007.

[24] 刘建中. 范德蒙行列式的一个性质的证明及其应用[J]. 河北大学学报(自然科学版), 2000, 20(1): 83-85.
LIU J Z. The proof of a property of Vandermonde determinant and applications[J]. Journal of Hebei University, 2000, 20(1): 83-85.

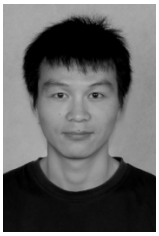
[25] HUANG C, XU L. Fast Software Implementations of Finite Field Operation[R]. Washington University, 2003.

[26] 董欢庆, 李战怀, 林伟. RAID-VCR: 一种能够承受三个磁盘故障的 RAID 结构[J]. 计算机学报, 2006, 29(5): 792-800.
DONG H Q, LI Z H, LIN W. RAID-VCR: A new RAID architecture for tolerating triple disk failures[J]. Chinese Journal of Computers, 2006, 29(5): 792-800.

作者简介：

陈钢（1982-），男，安徽淮南人，复旦大学博士生，主要研究方向为信息论与编码和并行计算。

朱俊峰（1977-），男，上海人，复旦大学博士生，主要研究方向为无线数据传输和移动通信。



张世乐（1980-），男，河南信阳人，硕士，诺基亚西门子通信有限公司工程师，主要研究方向为无线通信和移动通信。

吴百锋（1963-），男，上海人，硕士，复旦大学教授，主要研究方向为信息论与编码和嵌入式系统设计。