计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.181102161

基于 RBEC 的副本动态存储方法

洪海诚　陈丹伟
南京邮电大学计算机学院、软件学院、网络空间安全学院　南京 210023

摘　要　随着云存储技术的飞速发展,现有的云存储架构和存储模式都以一种静态的方式呈现在用户和攻击者面前,使得数据面临着更多的安全威胁。针对这种数据静态存储模式的不足,文中提出了一种基于二元随机扩展码(RBEC)的副本动态存储方法。该方法利用一种网络编码将数据块存储在云节点上,通过基于二元随机扩展码进行节点数据变换,可随机时变地改变节点的数据信息,通过变换攻击面来增加攻击者实施攻击的复杂度和成本,降低系统的脆弱性曝光和被攻击的概率,提高系统的弹性。理论分析和仿真实验结果表明,该方法对变换时的编码计算时间开销在整个动态变换中的占比不高,主要的时间开销是在节点间数据编码块的传输上。此外,文中还将该方法与一般再生码拟态变换方案做了性能对比分析。REBC 的特性,即重新生成的编码矩阵满足 MDS 性质的概率几乎为 1,所以文中所提方法的编码过程的性能开销优于一般再生码可能多次变换的性能开销。

关键词:副本存储;移动目标防御;存储编码;攻击面;二元随机扩展码
中图法分类号　TP393

Replica Dynamic Storage Based on RBEC

HONG Hai-cheng and CHEN Dan-wei
School of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210023,China

Abstract　With the rapid development of cloud storage technology,the existing cloud storage architectures and storage patterns are presented in a static way to users and attackers,making the data face more security threats. This paper proposed a duplicate dynamic storage scheme based on Random Binary Extension Code (RBEC). The scheme uses a network code to store the data blocks on the cloud nodes. The data information of the nodes can be changed randomly and time-varying by the node data transformation based on Binary Random Extension Code. By changing the attack surface,it can increase the complexity and cost of the attacker,reduce the vulnerability exposure and the probability of being attacked,and improve the flexibility of the system. The theoretical analysis and simulation results show that the coding computation time cost of this method is not high in the whole dynamic transformation,and its main time cost is the transmission of data encoding blocks between nodes. In addition,the performance of this method was compared with the general regenerative code mimetic transformation schemes. Because of the characteristics of REBC,that is,the probability of the regenerated encoding matrix satisfying the MDS property is almost 1,the performance overhead of this method is better than that of general regenerative code which may transform many times in the encoding process.

Keywords　Replica storage,Moving target defense,Network coding,Attack surface,Random binary extensive code

1　引言

在信息时代,数据存储安全一直是备受关注的问题。越来越多的用户将自己的数据托管到云存储系统[1],但现有的云存储系统基本将数据以静态存储模式存储在云节点中,这种存储模式极易曝光系统的脆弱性,增加了数据被攻击的概率,严重威胁到了用户的数据安全。

针对云存储系统的确定性和静态性带来的安全威胁,本文提出了一种移动目标数据存储方案。该方案将数据冗余存储在云节点中,然后通过对节点的随机时变来变换数据,以保证云存储系统的动态化和不可预测性;并且该方案支持数据的快速恢复和下载,保证了数据的完整性和可用性;通过二元随机扩展码来进行数据编码和译码,比传统的网络编码计算简单且传输带宽消耗小,在改变系统静态存储模式并增加系统安全性的同时降低了数据恢复和下载的成本。

2　相关工作

副本冗余存储[2]是目前保证数据可靠性的几大安全技术之一,本文主要研究纠删码[3]和再生码[4]的冗余存储[5-7]方案。目前主要的纠删码技术有 RS 码类、低密度奇偶校验码

到稿日期:2018-11-23　返修日期:2019-04-05　　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61602114);国家重点研发计划(2017YFB0801703)
This work was supported by the Key Program of the National Natural Science Foundation of China(61602114) and National Key R&D Program of China(2017YFB0801703).
通信作者:陈丹伟(chendw@njupt.edu.cn)

LDPC 和喷泉码，上述码类均可通过编码矩阵和译码矩阵进行描述。再生码基于网络编码理论，通过改进纠删码来减小节点的修复带宽，主要技术有最小存储再生码（Minimum Storage Regeneration，MSR）、最小带宽编码（Mininum Bandwidth Regeneration，MBR）以及基于这两种技术进行改进的相关再生码技术。文献[8]给出了 RS 码类在分布式存储中的精确修复的性能问题，研究表明，在某些参数条件下，RS 码类具有 MDS 这种最优再生码的线性修复性能。文献[9]提出了一种新型的半规则形式的 LDPC 码，这种 LDPC 码的奇偶校验矩阵由置换矩阵的子矩阵组成。文献[10]提出了一种用于分布式存储系统的可修复喷泉码的构造方案，这种编码在理论上是安全可用的，可以防止窃听者访问存储节点和下载其他节点数据来修复其他子集数据。文献[11]对再生码做了详细的综述，介绍了再生码的基本概念、单节点修复再生码和合作修复再生码的编码策略，最后总结了再生码的发展和研究方向。文献[12]提出了一种将最小存储再生码的外部码的设计和存储码的设计分离的通用方案，以保护用户数据免受窃听者通过访问网络链路或者单个节点的攻击。文献[13]提出了基于柯西矩阵构造的 MBR 码，其可以在不牺牲系统存储容量的情况下，保证数据块包含足够的数据信息，且不会泄露原始数据的任何信息。

以上的副本冗余存储技术大大提高了系统数据的可靠性，但并未改变系统的静态存储模式和确定的系统结构，这使得现有的防御策略都是根据攻击类型来确定的，也就是说防御方处于被动层面，这几乎很难防御现有的复杂多样且层出不穷的攻击手段，从而无法有效地保护数据安全。近年来提出的移动目标防御（Moving Target Defense，MTD）相关技术[14]是被称为改变计算机安全的游戏规则的技术。文献[15]首先阐述了 MTD 的背景知识，分析了 MTD 的概念，接着采用文献分析法阐述了 MTD 的设计原则和体系结构；此外，从生成策略、实行变换、性能评估等方面介绍了相关的关键技术，然后说明了 MTD 在不同网络体系结构中的应用；最后，阐述了该领域存在的问题和未来的发展趋势，为进一步的研究指明了方向。文献[16]介绍了 MTD 在网络中的应用，如 IP 地址随机化和端口地址随机化，以及网络层的网络结构和网络服务的随机化。文献[17]基于网络层的 MTD，提出了一种软件定义网络（Software Defined Network，SDN）的 MTD 方法，通过 SDN 的动态地址变换提高了系统的安全性。

本文综合副本冗余存储机制和移动目标防御思想，在网络编码的基础上引入了动态防御的技术，提出了动态副本存储方案。该方案增强了数据的安全性，并且提高了系统效率。

3 副本冗余与移动目标防御

本节主要介绍副本动态存储的相关原理和核心技术。副本动态存储的基本思路主要是在副本冗余存储的基础上，加入节点数据块的随机变换功能，使得节点数据的状态动态化，增加了攻击者攻击的复杂度和成本，提高了系统的安全性和可靠性。

动态变换功能是基于移动目标防御思想的。移动目标防御是一种为改变网络空间中“易守难攻”的局面所提出的设计指导思想，不是一种具体的防御方法，研究者可以将这种思想应用于系统的某一个属性上，以此来提高系统的安全性。

将数据存储在节点之前，需要对数据进行分块冗余处理，传统的处理方法是成熟的网络编码方式，即 RS 码。RS 码是一种基于有限域运算，参数选择自由，达到了理论上最优存储效率的编码方法。RS 码是 MDS 性质码，(n,k)RS 码需要将数据分为 k 份，然后通过 GF(2^m)有限域随机生成的编码矩阵将信息编码为 n 份编码数据块，在获取数据时只需要获取任意 k 份数据块就可以将原信息恢复，达到参数(n,k)下的最大容删能力。

由于 RS 码的构造及其译码运算始终无法脱离有限域的桎梏，当存储系统存储的数据规模逐渐增加时，RS 码的构造规模也需要增大，这使得有限域不断地增大，运算效率随之变低。为了解决这个问题，文献[15]提出了只需要简单做异或运算的二元随机扩展码编码，该编码是基于有限域 GF(2)运算的，编码矩阵的元素只有 0 和 1，其编码方式如图 1 所示。

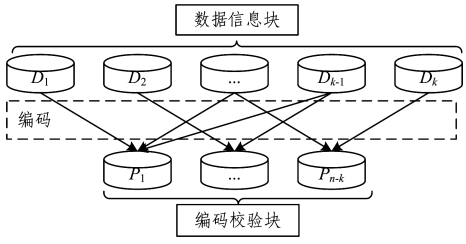


图 1 GF(2)有限域数据编码

Fig. 1 GF(2) finite field data coding

二元随机扩展码主要包括 n,k,δ,t 这 4 个参数，其中 n 为编码向量的编码长度，k 为信息部分的长度，δ 为保证高概率满秩所需的冗余位数，t 为可丢失的最大数据块数，它们之间存在如下关系：

$$n=k+\delta+t \tag{1}$$

采用的编码矩阵有 n 行、k 列，由单位矩阵和随机矩阵相结合的方式构成。

4 副本冗余动态存储机制

本节将对副本动态存储的过程进行详细介绍，该过程包括存储系统模型、数据上传与初始化、数据的下载与节点数据恢复。

存储系统模型是基于云存储架构展开的，该存储模型不考虑客户端节点，只有存储云节点以及一个存储数据编码信息的节点，如图 2 所示。

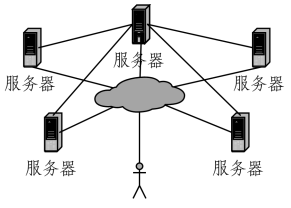


图 2 云节点存储模型

Fig. 2 Cloud node storage model

用户向可信云节点上传数据，所有的数据传输都是通过认证协议双向加密的，以保证数据在传输过程中不会被监听

和泄露。可信云节点的主要作用是在数据初始化时进行编码，生成编码向量，并在其后的变换中随机生成参数向量。其他云节点负责存储冗余数据和计算变换向量。

4.1 数据的上传与初始化

数据由用户从本地上传至云端，传输过程加密，数据是否需要在本地经过加密再上传，不属于本文的研究范围。为了保证数据时刻处于安全状态，会直接将数据上传至可信云节点位置，可信云节点不唯一。

数据的初始化步骤如下：

(1)可信节点生成一个 GF(2)域上的随机矩阵，其与单位矩阵结合构成编码矩阵；

(2)对数据块进行分块处理，数据均分为 k 块，最后一块大小不足时以 0 填充；

(3)通过编码矩阵对数据块进行编码，得到 n 个编码数据块，将这 n 个编码数据块存储在 n 个云节点中。

具体的初始化过程如下：

选取合适的参数 k,δ,t，在有限域 GF(2)中随机生成一个只包含 0,1 元素的矩阵 $\boldsymbol{R}_{(\delta+t)*k}$，与单位矩阵 $\boldsymbol{I}_{k*k}$ 结合构成编码矩阵 $\boldsymbol{G}_{n*k}=\begin{bmatrix}\boldsymbol{I}_{k*k}\\ \boldsymbol{R}_{(\delta+t)*k}\end{bmatrix}$。编码矩阵的结构如下：

$$\boldsymbol{I}_{k*k}=\begin{bmatrix}1&0&\cdots&0\\0&1&\cdots&0\\\vdots&\vdots&&\vdots\\0&0&\cdots&1\end{bmatrix} \tag{2}$$

$$\boldsymbol{R}_{(\delta+t)*k}=\begin{bmatrix}r_{1,1}&r_{1,2}&\cdots&r_{1,k}\\r_{2,1}&r_{2,2}&\cdots&r_{2,k}\\\vdots&\vdots&&\vdots\\r_{\delta+t,1}&r_{\delta+t,1}&\cdots&r_{\delta+t,k}\end{bmatrix} \tag{3}$$

为保证运算后的编码数据块具有足够的冗余度，即每一个信息数据块都尽可能地在校验数据块中有因子，随机生成的 $\boldsymbol{R}$ 应尽可能列满秩。虽然当行数足够多时，$\boldsymbol{R}$ 满秩的概率接近 1，但是需要做一个迭代判断，若 $\boldsymbol{R}$ 不满秩，则随机选取一行重新生成编码向量，直到满秩。

将原始数据进行 k 等分分块，并将分块数据记为矩阵 $\boldsymbol{M}_{k*1}$，将编码矩阵与数据块矩阵相乘可以得到编码数据块 $\boldsymbol{\beta}_{n*1}$，编码向量中的前 k 个行向量为单位矩阵，编码数据块可还原为原数据，后面的$(n-k)$个行向量为有限域 GF(2)的随机向量，是编码数据块的线性参数。整个运算公式如下：

$$\boldsymbol{\beta}_{n*1}=\boldsymbol{G}_{n*k}*\boldsymbol{G}_{k*1}=\begin{bmatrix}\boldsymbol{I}_{n*k}\\ \boldsymbol{R}_{(n-k)*k}\end{bmatrix}*\begin{bmatrix}D_1\\D_2\\\vdots\\D_k\end{bmatrix}=\begin{bmatrix}D_1\\D_2\\\vdots\\D_k\\P_1\\P_2\\\vdots\\P_{n-k}\end{bmatrix} \tag{4}$$

$$P_i=\sum_{j=1}^{k}r_{i+k,j}*m_{j,1} \tag{5}$$

其中，$i=1,2,\cdots,n-k$。

可信云节点将生成的 n 个编码数据块分别传输给 n 个云节点存储，每个节点存储一个数据块，存储前 k 个数据块的节点称为信息节点，后面 $n-k$ 个数据块称为校验节点，编码向量由可信云节点管理和维护。

4.2 数据的下载与恢复

数据下载是指用户下载数据时获得原数据，根据二元随机扩展码特性，随机选取 k 个线性不相关的行向量组成编码矩阵，即可恢复原数据。

数据下载时在可信云节点中随机选取 k 个线性不相关的行向量组成 $\boldsymbol{G}'_{k*k}$，计算该矩阵的逆矩阵 $\boldsymbol{G}'^{-1}_{k*k}$，然后左乘这些行向量对应的编码数据块 $\boldsymbol{\beta}_{k*1}$，即可得到原数据块矩阵 $\boldsymbol{M}_{k*1}$，最后将这些数据块合并。

数据的恢复是指当某一个云节点数据被误删或者云节点出现故障时，需要恢复该节点的数据。若该节点属于校验节点，则因为校验节点的编码参数本就是随机生成的，所以是否恢复校验数据块的影响不大。此时，为了节省计算开销，只需要重新随机生成一个编码参数向量，再从任意 k 个云节点获取编码数据块进行运算，即可得到新的校验数据块。若该节点属于信息节点，那么需要完整恢复数据块。假设该信息节点为存储数据块对应的第 i 行编码向量，则首先需要找到一个在 i 处为 1 的行向量，然后在 $\boldsymbol{R}_{(n-k)*k}$ 中选取包含这一向量的几行向量组成新的矩阵 $\boldsymbol{R}_{\infty*k}$，其与单位矩阵构成译码矩阵 $\boldsymbol{H}=\begin{bmatrix}\boldsymbol{R}^{\mathrm{T}}_{\infty*k}\\ \boldsymbol{I}_{\infty*\infty}\end{bmatrix}$。根据译码矩阵和编码数据块的关系：

$$H^{\mathrm{T}}_{(\infty+k)*k}*\boldsymbol{\beta}_{(\infty+k)*(n-k)}=0 \tag{6}$$

可以得到 k 个方程，这些方程的解集就是节点 i 的编码数据块。

4.3 数据块的动态变换

数据块的动态变换是指在数据初始化后，数据块存储在各个云节点中，为了改变云节点的静态存储模式，随机时间下随机选取某个节点进行数据块变换，以增加系统的复杂性，减小数据泄露的风险。

基于上一节的数据节点恢复，提出数据块的动态变换概念，整个过程由可信云节点来计算，主要步骤如下：

(1)随机选取一个节点 i，判断其是信息节点还是校验节点；

(2)如果是信息节点，那么随机选取一个校验节点来存储该信息节点的数据块，并将该校验节点的编码向量改为信息节点的编码向量；

(3)随机生成一个 $1*k$ 的行向量，从编码矩阵中随机选取 k 个行向量构成变换矩阵 $\boldsymbol{R}_{k*k}$，将该行向量与变换矩阵进行有限域 GF(2)相乘，得到新的编码参数行向量，同时将该行向量与对应的编码数据块矩阵相乘，得到新的编码数据块；

(4)如果该节点是校验节点，那么不需要执行步骤(2)，直接执行步骤(3)，改变该节点的数据块。

5 可用性证明与安全性分析

本节主要对副本冗余动态变换的可用性与安全性进行详细的分析与证明。

5.1 可用性证明

副本冗余及节点数据块的编码在变换后可以正确恢复原数据，即在编码向量的动态变换过程中不破坏数据的完整性

和可用性。本节对整个数据编码和编码参数变换进行详细分析。由于二元随机扩展码属于概率译码，因此须证明在任何时刻数据都可以高概率地译码成功。

定义 1 构成编码矩阵的 $\delta+t$ 行 k 列的随机矩阵 $\boldsymbol{R}$ 中的各元素 r_{ij} $(0\leqslant i<\delta+t, 0\leqslant j<k)$ 服从 0，1 分布，且各元素取值相互独立。

定义 2(可译码性) 如果在 n 个冗余数据块中选取 k 个数据块，并且根据对应的编码矩阵进行译码操作，最后可以还原出原始数据块，那么这些数据块具有可译码性。对于一个编码矩阵来说，其任意 k 行线性无关即表示其对应的编码数据块具有译码性。

引理 1 由单位向量 $\boldsymbol{I}$ 和随机矩阵 $\boldsymbol{R}$ 构造的编码矩阵 $\boldsymbol{G}_{n*k}$ 在丢失了任意 t 列数据之后，列满秩的概率的下界为：

$$P_{rank(G)=k}\geqslant 1-\frac{1}{2^{\delta}} \tag{7}$$

证明略，详情请参考文献[18]。

定理 1 数据块均分为 k 块后，将其通过二维随机扩展码所构造的编码矩阵编码成 n 块，然后选取任意 k 块数据，并根据选取的数据块构造译码矩阵 $\boldsymbol{H}$，从而成功译码出原数据块。

证明：根据引理 1，对于二元随机扩展码所构造的编码矩阵，在缺失 t 列或少于 t 列的数据之后，其对应的编码矩阵满秩的概率下界为 $1-\frac{1}{2^{\delta}}$，也就是说，当 δ 取足够大时，编码矩阵满秩的概率无限接近 1，因此基本可以认为编码矩阵满秩。

根据定义 2，在丢失了 t 列数据后，编码向量高概率列满秩，编码向量依旧保证了数据块的可译码性，即可以成功还原出原数据块。

定理 2 编码数据块在经过多次动态变换后依然具有可译码性。

证明：由于每次数据块的动态变换都是随机地选取一个节点进行变换，假设编码数据有 n 块，其中信息节点 k 个，校验节点 $n-k$ 个，且未有节点数据丢失(若有数据块丢失，则可以先恢复)。

根据定理 1，节点数据块在动态变换之前，若未缺失任何数据，其编码矩阵是列满秩的，因此这些编码数据块是具有可译码性的。

假设在第 l 次变换后编码矩阵列满秩，那么在第 $l+1$ 次变换时，由于每次变换都是随机选取节点 i，如果节点 i 是信息节点，那么随机选取一个校验节点 j 来存储节点 i 的数据块，且编码向量随之更换，再随机生成编码行向量进行编码运算，具体如图 3 所示。

由图 3 可以看出，在变换前后，编码矩阵变化的只有随机矩阵 $\boldsymbol{R}$ 的第 j 行数据，而第 j 行数据是随机生成的，根据定理 1，变换后的编码矩阵 $\boldsymbol{G}'$ 满足列满秩，因此编码数据块具有可译码性。同理，若选取的节点为校验节点，那么根据 4.3 节的数据块变换介绍，这里只需要重新生成该行的编码向量，因此去掉图的中间一部就可以表示这种情况，与信息节点相同，该编码矩阵列满秩。

$$\boldsymbol{G}_{n\times k}=\begin{bmatrix}
1 & 0 & \cdots & 0 & \cdots & 0\\
0 & 1 & \cdots & 0 & \cdots & 0\\
\vdots & \vdots & & \vdots & & \vdots\\
i\{0 & 0 & \cdots & 1 & \cdots & 0\}\\
\vdots & \vdots & & \vdots & & \vdots\\
r_{1,1} & r_{1,2} & \cdots & r_{1,i} & \cdots & r_{1,k}\\
r_{2,1} & r_{2,2} & \cdots & r_{2,i} & \cdots & r_{2,k}\\
\vdots & \vdots & & \vdots & & \vdots\\
r_{j,1} & r_{j,2} & \cdots & r_{j,i} & \cdots & r_{j,k}\\
\vdots & \vdots & & \vdots & & \vdots\\
r_{n-k,1} & r_{n-k,2} & \cdots & r_{n-k,i} & \cdots & r_{n-k,k}
\end{bmatrix}$$

j 行编码向量替换为 i 行

i 行编码向量随机生成

$$\boldsymbol{G}'_{n\times k}=\begin{bmatrix}
1 & 0 & \cdots & 0 & \cdots & 0\\
0 & 1 & \cdots & 0 & \cdots & 0\\
\vdots & \vdots & & \vdots & & \vdots\\
r'_{j,1} & r'_{,2} & \cdots & r_{j,i} & \cdots & r'_{j,k}\\
\vdots & \vdots & & \vdots & & \vdots\\
0 & 0 & \cdots & 0 & \cdots & 1\\
r_{1,1} & r_{1,2} & \cdots & r_{1,i} & \cdots & r_{1,k}\\
r_{2,1} & r_{2,2} & \cdots & r_{2,i} & \cdots & r_{2,k}\\
\vdots & \vdots & & \vdots & & \vdots\\
0 & 0 & \cdots & 1 & \cdots & 0\\
\vdots & \vdots & & \vdots & & \vdots\\
r_{n-k,1} & r_{n-k,2} & \cdots & r_{n-k,i} & \cdots & r_{n-k,k}
\end{bmatrix}$$

调整 i、j 行编码向量位置

$$\boldsymbol{G}'_{n\times k}=\begin{bmatrix}
1 & 0 & \cdots & 0 & \cdots & 0\\
0 & 1 & \cdots & 0 & \cdots & 0\\
\vdots & \vdots & & \vdots & & \vdots\\
i\{0 & 0 & \cdots & 1 & \cdots & 0\}\\
\vdots & \vdots & & \vdots & & \vdots\\
0 & 0 & \cdots & 0 & \cdots & 1\\
r_{1,1} & r_{1,2} & \cdots & r_{1,i} & \cdots & r_{1,k}\\
r_{2,1} & r_{2,2} & \cdots & r_{2,i} & \cdots & r_{2,k}\\
\vdots & \vdots & & \vdots & & \vdots\\
r'_{j,1} & r'_{j,2} & \cdots & r_{j,i} & \cdots & r'_{j,k}\\
\vdots & \vdots & & \vdots & & \vdots\\
r_{n-k,1} & r_{n-k,2} & \cdots & r_{n-k,i} & \cdots & r_{n-k,k}
\end{bmatrix}$$

图 3 信息节点变换矩阵的变化过程

Fig. 3 Change process of matrix in information node transformation

综上所述，在数据节点基本完整即及时修复丢失节点数据的情况下，数据节点的动态变换过程并不影响编码矩阵的列满秩属性，也就是说编码数据块基本具有可译码性。

5.2 安全性分析

副本冗余动态存储方案主要从以下几个方面提高数据的安全性。

(1)副本冗余性。副本冗余技术是为了保证数据可靠性而提出的，通过增加数据存储空间来换取数据的高可靠性。将本文数据分为 k 块后进行 RBEC 编码，根据随机编码向量将数据编码为 n 个数据信息块，并将其存储在 n 个节点中；在

攻击者攻击之后，丢失的数据块不超过 t 个，这些数据都可以通过编码矩阵和剩余的数据块进行恢复。副本冗余性通过避免节点损毁造成的数据丢失，来保证数据的完整性和可用性，以增强系统的可靠性。

(2)变换随机性。在冗余数据块存储的节点中，采用随机操作进行动态变换以增加系统的随机性。首先，每一次变换的时间间隔 T 都是随机决定的，但在一定范围内，T 太大会导致变换的意义较小，T 太小会影响系统的性能；其次，随机选取变换目标节点 i，而且在节点 i 为信息节点的情况下，校验节点 j 也是随机选取的；最后，编码向量也是随机产生的，随机选取 k 个节点的编码向量和数据块进行矩阵运算。攻击者可以较轻松地攻击一个静态的系统，但对于随机性较强的系统，攻击者的攻击难度会大大增加，从而使得系统的安全性和可靠性得到明显的提升。

(3)系统攻击面变换机制。副本冗余动态存储方法是根据移动目标防御机制提出的，其目的是改变系统的静态存储模式，提高系统的复杂度和安全性，从而增加攻击者的攻击难度和成本。由于针对每一个系统漏洞进行修复的难度较高，因此通过改变系统存储状态，改变或减小系统的攻击面，来使得攻击者的攻击难以维持，从而有效地保护数据的安全。

6 实验与分析

本节针对副本冗余动态存储的正确性和可靠性进行详细的仿真实验，实验环境的配置为 Intel(R)Core(TM)i5-2450M CPU @ 2.50GHz 核处理器和 6GB RAM，Ubuntu14.04 操作系统。仿真实验使用的是 Python 语言，借助 numpy 包进行矩阵运算和随机生成编码参数。本实验简单实现文件的上传、下载和动态变换，对编码数据块动态变换的性能进行分析。

该实验主要对动态变换的两个方面做了详细的分析，一是二元随机扩展码的参数对编码数据块的变换性能；二是文件分块大小对编码数据块变换的带宽消耗和性能。

仿真实验实现了以下几个过程。

(1)数据的上传

用户将数据上传至可信云节点，数据传输时间不计入时间开销。云节点将数据分块，并生成随机编码向量。

云节点对数据块编码，然后将生成的编码数据块传输给各个节点。

(2)信息数据块的修复

信息节点数据丢失时，需要包含该数据块因子的校验数据块来计算恢复数据，因此先选取该位置为 1 的行向量。

选取包含该向量的部分校验编码向量，然后与单位向量构成译码矩阵。

根据译码矩阵与编码数据块的关系，构造多组等式方程(该方程理论上可以恢复多个节点数据，这里只以单节点为例)，方程组的解集即为丢失节点的信息数据块。

(3)数据块的变换

在随机的时间条件下，随机选取一个存储节点，判断该存储节点的类型。该节点如果是信息节点，那么进行节点数据交换，然后对该节点生成随机编码向量；如果是校验节点，那么直接生成随机编码向量。从剩下的节点中随机选取 k 个节点，将编码数据块传输至可信云节点处。

将新的编码向量与选出的 k 阶编码矩阵相乘，得到新的节点编码向量；同时，新的编码向量需要与 k 个数据块列向量相乘，从而得到新的编码数据块。

(4)数据的下载

随机选取 k 个节点，并且判断这 k 个节点对应的 k 阶矩阵是否可逆，如果不可逆，那么需要重新选取节点，直至矩阵可逆。计算 k 阶矩阵的逆矩阵，将其与编码数据块列向量相乘，从而得到原数据块列向量。将原数据块按分块信息合并，从而得到原数据。

6.1 动态变换中 RBEC 的参数

鉴于二元随机扩展码各参数之间具有 $n=k+\delta+t$ 的关系，本实验只对参数 k，δ，t 进行实验讨论。k 作为数据分块信息码数，分别取值为 10，20，40；δ 是保证编码矩阵高概率满秩的参数，分别取值为 5，10，20；t 是最大可丢失数据数，分别取值为 5，10，20。此外，为保证数据高概率恢复，另取限制条件 $t<k$，即丢失节点数不超过信息码节点数。按照以上参数取值及限制条件，最后整理出 7 组数据，如表 1 所列。

表 1 不同编码参数的各项时间开销

Table 1 Time cost of different coding parameters

(单位：s)

序号	$k/\delta/t$	上传时间/s	下载时间/s	节点修复时间/s
1	10/20/10	220.75	165.06	93.40
2	20/5/10	221.15	165.54	93.66
3	20/10/10	221.23	165.65	93.85
4	20/20/10	221.94	165.88	94.31
5	40/20/5	223.14	165.97	94.72
6	40/20/10	223.49	166.14	94.81
7	40/20/20	223.22	166.12	94.87

表 1 列出了在带宽为 100 Mbp 的条件下，对 1 GB 文件进行 100 次上传、下载以及单节点数据修复的平均时间。从表中可以看出，无论哪个参数变大，都会使上传、下载节点的修复增加一些时间开销，而编码和译码所花费的时间开销对整体性能开销的影响不大。表 1 中的时间开销大部分是数据传输的时间开销：首先将数据从本地上传至云端，下载数据时，将编码数据块传输至可信云节点，然后将原数据从云端下载到本地，当节点恢复时，将编码数据块传输至本地，最后将修复的编码数据块传回节点。

由表 2 可得，即使 δ 取 5，变换成功率也高达 0.96，而且成功率随着 δ 的增加而上升，δ 取 10 时变换成功率已经达到 0.99。变换时间比单节点修复时间略长，但比数据上传时间和下载时间短很多，这主要是因为节点数据块变换都是在可信云节点上进行的，这期间需要判断该云节点是否为信息节点，然后进行单节点数据传输，即选取 k 个节点，将这些节点的数据传输到可信云节点处，再进行编码和译码，最后还要将生成的数据块传回给变换节点，因此时间开销会比单节点修复的时间开销大。

表 2 不同编码参数的变换开销

Table 2 Transform cost of different coding parameters

序号	$k/\delta/t$	变换成功率	平均变换时间/s
1	10/20/10	0.99	95.44
2	20/5/10	0.96	98.34
3	20/10/10	0.99	97.95
4	20/20/10	0.99	97.58
5	40/20/5	1.00	99.76
6	40/20/10	0.99	99.49
7	40/20/20	1.00	98.97

由表 1、表 2 可以知道，除去数据传输开销外，编码和译码的计算开销以及编码矩阵的生成和计算所占用的时间都不长，而且 k 越小则时间开销越小，δ 越大则变换成功概率越高，t 对计算开销和成功率的影响不大，只是能确保当前数据节点丢失的最大值。因此，系统应在合理的性能开销范围内选取 k 和 δ 的值。

6.2 动态变换中数据的大小

6.1 节讨论了在 1 GB 数据文件下不同参数对节点数据块动态变换性能的影响，本节讨论在固定参数 $k=20$，$\delta=10$，$t=10$ 以及带宽为 100 Mbp 的条件下不同数据大小对变换性能的影响，数据大小分别选取 100 MB，200 MB，400 MB，800 MB 和 1 GB。实验结果如表 3 和表 4 所列。

表 3 不同数据大小时的各项时间开销

Table 3 Time cost of different data sizes

数据大小/ M	上传时间/s	下载时间/s	单节点修复时间/s
100	23.71	16.17	8.57
200	49.43	30.41	17.10
400	99.91	61.53	38.28
800	200.37	125.00	77.65
1024	221.23	165.65	93.85

表 4 不同数据块的变换时间开销

Table 4 Transform cost of different data sizes

数据大小/ M	变换成功率	平均变换时间/s
100	1.00	9.31
200	0.99	19.63
400	0.99	39.28
800	1.00	79.30
1024	0.99	97.95

由表 3、表 4 可以看出，随着数据变大，整个系统的开销不断变大，数据的传输依然占据了很大一部分时间开销；但是从这两张表中无法直观地看出数据大小对变换过程中的编码性能的影响，因此图 4 给出了条形图来清晰表示动态变换过程中数据传输、数据块重编码以及编码数据块传回的变化情况。

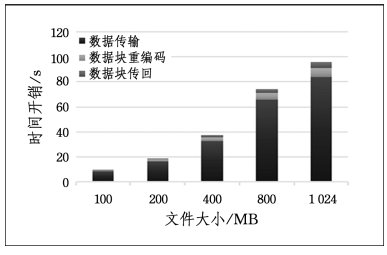


图 4 不同数据大小变换的时间开销

Fig. 4 Time cost of different data sizes

由图 4 可以看出，随着数据大小的增加，动态变换时间也逐渐增加，并且如上文的分析，时间开销主要集中在数据传输方面，而数据块重编码的时间开销虽然随着数据的增加而增加，但在整个变换过程中一直只占很小的一部分。因此，总的来说，在整个副本冗余动态存储中，需要根据系统情况做好参数的选取，以及动态变换时间间隔范围的取值，这样整个系统的性能才会达到最佳，在不对系统资源造成浪费的同时提高系统的安全性和可靠性。

6.3 与普通再生码拟态存储方案的性能对比

本节将本文方法与现有的一般再生码拟态存储方案[19]进行性能对比分析。文献[19]中的方法将数据上传至服务器，进行分块编码，然后按以下步骤进行拟态变换：

1)编码初始化生成 $n(n-k)\times k(n-k)$ 大小的编码矩阵，矩阵满足 MDS 性质；

2)拟态变换，随机选取一个节点，令其对应的 $(n-k)$ 个行向量为 0；

3)随机生成一个大小为 $(n-k)\times(n-1)$ 的变换矩阵；

4)从剩下的每个节点中取一个编码向量组成一个大小为 $(n-1)\times k(n-k)$ 的矩阵；

5)将步骤 3)和步骤 4)的矩阵相乘得到新的矩阵，并用其代替要变换的节点编码数据块；

6)若新组成的编码矩阵满足 MDS 性质，则变换成功，否则转步骤 3)。

根据上文的实验数据，这里的参数仍然取 $n=40$，$k=20$，$\delta=10$，$t=10$ 且带宽为 100 Mbp，数据大小分别取为 100 MB，200 MB，400 MB，800 MB 和 1 GB。实验主要对比一般再生码和 REBC 再生码完成一次变换的编码时间开销，具体实验结果如图 5 所示。

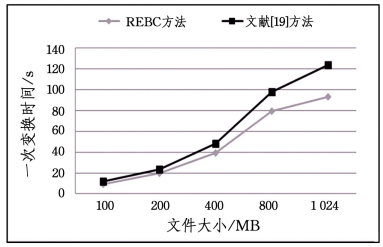


图 5 一次变换时间开销的对比图

Fig. 5 Time cost contrast graph of one-time mimetic transform

从图 5 和 6.2 节的实验可以看出，文献[19]中的方案和本文方法的时间开销主要都是在数据传输上，在一个节点上的一次完整变换需要在待变换节点和运算服务器之间进行数据编码块传输。除此之外，本文方法在参数 $n=40$，$k=20$ 时有一定的性能优势，因为此时文献[19]中的方法在有限域 GF(28)下的一次变换成功率较低，往往需要进行 3 次或 3 次以上的变换才能成功，而本文所用的二元随机扩展码是在有限域 GF(2)下随机生成编码译码矩阵，满足 MDS 性质的概率几乎为 1，因此在编码计算上有一定的优势。

结束语 本文针对现有的云存储模式的静态特性、易被攻击者挖取漏洞并针对漏洞进行攻击的情况，提出了一种基于二元随机扩展码的动态副本存储方案。该方案通过二元随机扩展码构造的编码矩阵对分块数据块进行了编码运算，然后进行分布式存储，将关键信息存储在可信云节点中，接着随机时变地在可信云节点中进行动态编码数据块变换，利用移

动目标防御机制改变现有系统的状态,提升了攻击者攻击成功的难度和成本。文中给出了该方案的可用性证明和安全性分析,详细介绍了该方案的理论基础和现实依据;最后对该方案进行了仿真实验,针对二元随机扩展码的编码参数和数据文件大小分别进行了时间和性能开销的仿真。实验结果表明,在不考虑系统数据传输和 IO 开销的情况下,冗余动态变换的计算开销较低,所提方案的可行性较高,能够保证数据的安全性和系统的可靠性。

参 考 文 献

[1] CHEN H C H,HU Y,LEE P P C,et al. NCCloud:A network-coding-based storage system in a cloud-of-clouds[J]. IEEE Transactions on computers,2014,63(1):31-44.

[2] LEONTIADIS I,CURTMOLA R. Secure Storage with Replication and Transparent Deduplication[C]// Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy. ACM,2018:13-23.

[3] SHAHABINEJAD M,ARDAKANI M,KHABBAZIAN M. An erasure code with reduced average locality for distributed storage systems[C]// 2017 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2017: 427-431.

[4] GLIGOROSKI D,KRALEVSKA K,JENSEN R E,et al. Repair Duality with Locally Repairable and Locally Regenerating Codes [C]// 2017 IEEE 15th Intl Dependable, Autonomic and Secure Computing,15th Intl Conf on Pervasive Intelligence & Computing,3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE,2017:979-984.

[5] MAHAJAN J,BORKAR B S. Public Auditing for Regenerating Code Based Cloud Storage[C]// International Conference on Science and Engineering for Sustainable Development. IEEE, 2017:16-22.

[6] WARAMBHE P,GODE S,MULE S. Privacy-Preserving and Public Auditing for Regenerating-Code-Based Cloud Storage Using Finger Print Authentication[C]// International Conference on Communication & Electronic Information Engineering. IEEE,2017:153-166.

[7] SURYAWANSHI M,KAMBLE A,HIREKHAN S,et al. Privacy-Preserving and Public Auditing for Regenerating-Code-Based Cloud Storage[C]// International Conference on Communication & Electronic Information Engineering. IEEE,2018:763-769.

[8] GURUSWAMI V,WOOTTERS M. Repairing reed-solomon codes[J]. IEEE Transactions on Information Theory, 2017, 63(9):5684-5698.

[9] JIANG S,MO F,LAU F C M,et al. Tree-Permutation-Matrix Based LDPC Codes[J]. IEEE Transactions on Circuits and Systems II:Express Briefs,2018,65(8):1019-1023.

[10] KUMAR S,ROSNES E,I AMAT A G. Secure repairable fountain codes[J]. IEEE Communications Letters, 2016, 20(8): 1491-1494.

[11] HAO J,LU Y B,LIU X J,et al. Survey for regenerating codes for distributed Storage[J]. Journal of Chongqing University of Posts and Telecommunications(Natural Science Edition),2013, 25(1):30-38.

[12] KADHE S,SPRINTSON A. Universally weakly secure coset coding schemes for minimum storage regenerating (MSR) codes [C]// 2017 55th Annual Allerton Conference on Communication,Control,and Computing (Allerton). IEEE,2017:25-30.

[13] DAU S H,SONG W,YUEN C. On block security of regenerating codes at the MBR point for distributed storage systems [C]// 2014 IEEE International Symposium on Information Theory (ISIT). IEEE,2014:1967-1971.

[14] EVANS D,NGUYEN-TUONG A,KNIGHT J. Effectiveness of moving target defenses[M]// Moving Target Defense. New York:Springer,2011:29-48.

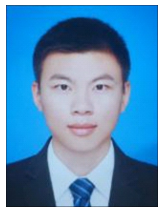
[15] LEI C,ZHANG H Q,TAN J L,et al. Moving Target Defense Techniques:A Survey[J]. Security and Communication Networks,2018(2):14-27.

[16] CARVALHO M,FORD R. Moving-target defenses for computer networks[J]. IEEE Security & Privacy,2014,12(2):73-76.

[17] WANG S,ZHANG L,TANG C. A new dynamic address solution for moving target defense[C]// Information Technology, Networking, Electronic and Automation Control Conference. IEEE,2016:1149-1152.

[18] CHEN L,ZHANG J Z,TENG P G,et al. Random Binary Extensive Code:An Efficient Code for Distributed Storage System [J]. Chinese Journal of Computers,2017,40(9):1980-1995.

[19] CHEN Y,WANG L J,YAN X C,et al. Mimic storage scheme based on regenerated code [J]. Journal on Communications, 2018,39(4):21-34.



HONG Hai-cheng, born in 1994, master. His main research interests include network security and network dynamic defense technology.



CHEN Dan-wei, born in 1970, Ph. D, professor. His main research interests include Virtualization and cloud technologyand Computer communication network and security.