Acta Technica 62 (2017), No. 6B, 957–970 © 2017 Institute of Thermomechanics CAS, v.v.i.

Remote data cloud storage algorithm based on bifurcation variational filtering

Hongqing Liu[1], Yan Liu[2], Diqing Shu[3], Zhixian Zou[1]

Abstract. In the application of cloud storage, user file is not in local storage, therefore, file security, data confidentiality and robustness are the key problems. Firstly, aimed at the secure erasure code storage system of several key servers proposed by existing literatures, the problem that data robustness leads to defects in data recovery is not considered, and the end-to-end checking strategy threat model of cloud storage is established by using pseudo-random linear mapping; secondly, interface file block structure is prepared and integrity check project design is conducted by referring to relevant literature algorithm to realize the supplement of secure erasure code storage system algorithm function of multiple key servers and provide the analysis of calculation complexity of algorithm; at last, the experiment result shows that, the proposed integrity check program can realize greater successful retrieval probability of data.

Key words. Cloud storage, Data security, Bilinear mapping, Confidentiality, Pseudo-random .

1. Introduction

Cloud storage system is driven by high speed network and big data center and can provide reliable storage service on network. Users can store files on cloud storage server and later visit [1] through internet. In file storage process, the most concerned matter for users is the security of stored files, of which data confidentiality and robustness are the key security problems to be solved. In order to guarantee the data confidentiality, users can encrypt files at the first and then store confidential files in cloud storage. Therefore, cloud file confidentiality is not only oriented at

[1]Hunan Vocational College of Modern Logistics, Changsha, Hunan, 410131, China
[2]Hunan Mechanical & Electrical Polytechnic, Changsha, Hunan, 410151, China
[3]Hunan Provinceial Research Institute Of Education, Changsha, Hunan, 410005

external personnel, but also oriented at cloud service provider [2]. Data robustness mainly involves two aspects: service failure and service damage. Service failure is that users fail to obtain request response from retrieval and service damage is that users obtain damaged data. The Paper mainly considers data robustness problem [3].

In literature [4, 5], cloud storage system is proposed to ensure data confidentiality. The system is a distributed storage system based on secure erasure of code, which is called as Securities and Exchange Commission (SEC) storage system. It includes storage server and key server. Files can be coded in ciphertext through distributed secure erasure code and the ciphertext will be randomly assigned to storage server for combination and storage. Each storage server will combine the received ciphertext into independent one. According to the characteristics of distributed secure erasure code, the files are confidential and can be recovered as long as there are enough numbers of storage servers and correct ciphertext return.

2. System plan and threat model

2.1. System scheme

Hybrid cloud environment is taken into account. Assuming that there is storage server $(SS_1, SS_2, \cdots, SS_n)$ in public cloud, there are m groups of key servers $(KS_1, KS_2, \cdots, kS_m)$ in private cloud. See Fig. 1 for the system plan. There are 4 stages in storage system: setup stage, storage stage, integrity check stage and retrieval stage.

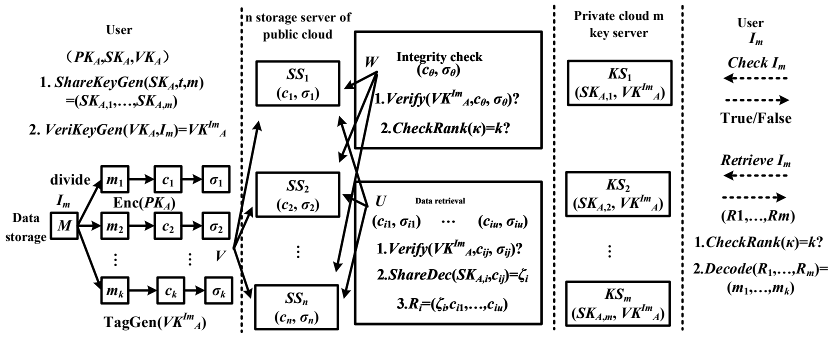


Fig. 1. System plan

At setup stage, system manager publishes system parameters. User A generates public key tuple (PK_A, SK_A) and master verification key VK_A. User A can share its key in key server through threshold value (t, m), for example, share key $SK_{A,i}$ in key server KS_i, where $1 \leq i \leq m$.

At storage stage, User A endows file M with identifier I_M and then User A uses master verification key VK_A to calculate the verification key $VK_A^{I_M}$ for file M. for the storage of file M, User A divides M into m_i file blocks $(m_1, m_2, \cdots, m_k)$ and then uses public key PK_A to change file block m_i into ciphertext c_i, and at the same time, User A generates complete label σ_i by using verification key $VK_A^{I_M}$ as the ciphertext c_i. After that, User A will assign ciphertext- label pair to v random storage servers, of which, after receiving tuples, each one SS_i will select combination coefficient vector $\overrightarrow{G}_j = (g_{j,1}, g_{j,2}, \cdots, g_{j,k})$ randomly and store the results as coding tuple (c_i', σ_i'). Coding tuple $\{(c_i', \sigma_i')\}$ and ciphertext $\{(c_i')\}$ are called as linearly independent only when its combination coefficient vector $\{\overrightarrow{G}_j\}$ is linearly independent of each other.

At integrity check stage, User A endows file M with identifier I_M and sends integrity check request to key server KS_θ, of which $\theta \in \{1, 2, \cdots, m\}$. After receiving the request, KS_θ inquires ω storage servers randomly for ω coding tuples $\left\{\left(c_{\theta_i}', \sigma_{\theta_i}'\right)\right\}_{i=1}^{\omega}$, of which $\omega > k$. After that, KS_θ integrates query results to get $(c_\theta', \sigma_\theta')$ and then checks its integrity by using verification key. If the verification fails, KS_θ will conducts non-intersection groups on $\left\{\left(c_{\theta_i}', \sigma_{\theta_i}'\right)\right\}_{i=1}^{\omega}$ to check the integrity of each group until all the damaged tuples are found. And at the same time, KS_θ will also check the translatability of correct ciphertext. Given that $\kappa = \left[g_{\theta_{i,j}}\right]$ is the combination coefficient matrix of correct ciphertext $\left\{c_{\theta_i}'\right\}$, of which $1 \le j \le k$. And the correct ciphertext $\left\{c_{\theta_i}'\right\}$ will be translatable only if and only if $Rank\,(\kappa) = k$. At last, KS_θ will returns the integrity check result and inform User A of the damaged storage service.

At retrieval stage, User A endows file M with identifier I_M and sends retrieval request to key server KS_i, and the later one will inquire u services randomly for u coding tuples after receiving the request. After that, KS_i will check the integral unit of tuple and filtrate the damaged ones. KS_i generates decryption token ζ_i in remained ciphertext by using shared key $SK_{A,i}$, and then returns the result $R_i = \left(\zeta_i, c_{i_1}', c_{i_2}', \cdots, c_{i_u}'\right)$ from key server, checks its translatability and decodes it to realize the reconstruction of k file blocks $(m_1, m_2, \cdots, m_k)$. If there are k ciphertexts are linearly independent in $(R_1, \cdots, R_m)$, above decoding process will be effective.

2.2. Threat model

Data robustness shows the successful retrieval of data. It needs correct ciphertext and enough quantities of storage servers. And integrity check program can be used to treat the damage problems of storage server. If storage server fails and its damage proportion is below the preset threshold value ε_0, data can be recovered by secure dispersion erasure code. Securities trading storage system parameters depend on the threshold value ε_0 of system. Security of integrity check program can be constructed by the security game of faking integrity label of opponents. See Fig. 2 for security game, which is defined as follows [10-11]:

Challenger C plays the role of user while opponent A plays the role of storage server. A inquires the integrity label of selected ciphertext and tries to fake the effective tuples of ciphretext and integrity label to win the security game (see Fig. 2), and the detailed process is as follows:

Step 1: at setup stage, C generates system parameter π and public key tuple (PK, SK) and verifies VK for target users and then sends (π, PK) to A.

Step2: at inquiry stage, A selects file block $(m_1, m_2, \cdots, m_k)$ randomly as the inquiry of C. Then C encrypts each file block m_i as ciphertext c_i with public key PK and use VK to generate integrity label σ_i. C feeds $\{(c_i, \sigma_i)\}_{i=1}^{k}$ back to A.

Step 3: at response stage, A tries to construct effective tuple (c^*, σ^*) with combination coefficient vector $\overrightarrow{G}^* = (g_1^*, g_2^*, \cdots, g_k^*)$. If (c^*, σ^*) passes verification, it will be effective, namely $Verify(VK, c^*, \sigma^*, \overrightarrow{G}^*) = Valid$. If (c^*, σ^*) is effective and is not $\{(c_i, \sigma_i)\}_{i=1}^{k}$ combination, then A wins the security game, namely:

$$(c^*, \sigma^*) = \left(\prod_{i=1}^{k} c_i^{*g_i^*}, \prod_{i=1}^{k} \sigma_i^{*g_i^*}\right) \neq \left(\prod_{i=1}^{k} c_i^{g_i^*}, \prod_{i=1}^{k} \sigma_i^{g_i^*}\right). \tag{1}$$

Therefore, A fakes an effective ciphertext markup tuple (c_i^*, σ_i^*) at least. And the advantage of opponent A can be defined as: $\Pr[\mathcal{A}]$.

Definition 1: an integrity check program is (t, ε) safe, if there is no such probabilistic algorithm making A win the security game within the time t.

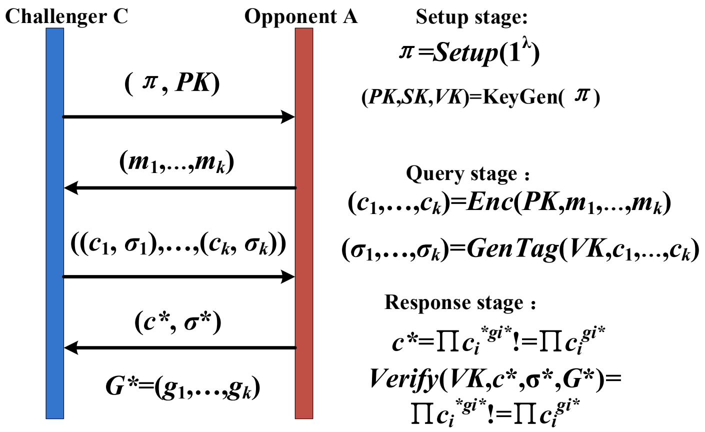

Challenger C
Opponent A
Setup stage:
π=Setup(1^λ)
(PK,SK,VK)=KeyGen(π)
(π, PK)
(m_1,...,m_k)
Query stage :
(c_1,...,c_k)=Enc(PK,m_1,...,m_k)
((c_1, σ_1),...,(c_k, σ_k))
(σ_1,...,σ_k)=GenTag(VK,c_1,...,c_k)
Response stage :
(c*, σ*)
$c^*=\prod c_i^{*gi*}!=\prod c_i^{gi*}$
G*=(g_1,...,g_k)
Verify(VK,c*,σ*,G*)=
$\prod c_i^{*gi*}!=\prod c_i^{gi*}$

Fig. 2. Security game integrity check program

3. Integrity check program

3.1. Construction of interface file block

Step 1 (bilinear map): given that $G = \langle g \rangle$ and $G_T = \langle g_T \rangle$ are the prime order double-cyclic multiplier clusters and the subscript T in the formula denotes target. Bilinear mapping $\hat{e} : G \times G \to G_T$ satisfies characteristics [12].

Bilinearity: $\forall x, y \in Z_p$, $\hat{e}(g^x, g^y) = \hat{e}(g, g)^{xy}$; non-degeneration: $\hat{e}(g, g) = g_T$; computability: $\hat{e}(\cdot, \cdot)$ can be calculated based on probabilistic polynomial time algorithm.

Step 2 (pseudo-random function): the random function from domain D and interval R can be selected from all the possible existing functions from domain D and interval R. On the other hand, pseudo-random function is selected from relatively small functions family. Therefore, pseudo-random function can be instantiated through functions family with relatively small index. It is hard to distinguish output distribution of pseudo-random function from that of random function [13].

Definition 2: given that $F : I \times D \to \mathcal{R}$ is random function, of which I is index space, $\mathcal{D}$ is input space and $\mathcal{R}$ is output space. And at the same time, under the circumstance that there is no probabilistic algorithm available distinguishing output distribution advantages within the time t, define the function $F' : I' \times D \to \mathcal{R}$ as the pseudo-random function, in which $I' \in I$ and $|I'| \ll |I|$.

Step 3 (dispersion erasure code): use dispersion erasure code which is the sparse generator matrix of random linear codes, of which the used linear combination in coding and decoding process is prime order p cyclic multiplier G_T. The dimensionality of generated matrix $G = [g_{i,j}]$ is $n \times k$, of which $g_{i,j} \in_R Z_p$. The construction of G contains n random selection linear independence vectors $\{\boldsymbol{G_i} = (g_{i,1}, g_{i,2}, \cdots, g_{i,k})\}_{i=1}^{n}$. Then k file blocks $(m_1, m_2, \cdots, m_k) \in G_T^k$ is coded as n symbols $(\omega_1, \omega_2, \cdots, \omega_n) \in G_T^n$, of which:

$$\omega_i = \prod_{j=1}^{k} m_j^{g_{i,j}} . \tag{2}$$

As for decoding, decoder shall collect k symbols $(\omega_{l_1}, \omega_{l_2}, \cdots, \omega_{l_n})$ at least. Given that the inverse matrix of $k \times k$ matrix constructed by $(\omega_{l_1}, \omega_{l_2}, \cdots, \omega_{l_n})$ due to $\kappa = [g_{l_i,j}]_{1 \le i,j \le k}$ is $\kappa^{-1} = [\hat{g}_{l_i,j}]_{1 \le i,j \le k}$. If κ is reversible, the decoding process will be:

$$m_i = \prod_{j=1}^{k} \omega_{l_j}^{\hat{g}_{i,j}} . \tag{3}$$

Step 4 (public key decryption): it consists of six algorithms including *Setup*, *KeyGen*, *KeyGen*, *Enc*, *ShareDec* and *FinalDec*. Where, *Setup* generates public parameters of system, $KeyGen$ generates public key tuple (PK, SK) for users, $ShareKeyGen$ decomposes secrete key SK of users into m shared keys, in which way users can recover data at any time t. Enc encrypts files as the ciphertext with public key PK. $ShareDec$ transforms ciphertext through the sharing of secrete key. $FinalDec$ inputs partial decoded ciphertext and then outputs the file.

3.2. Algorithm construction

Step 1 (setup stage): system manager uses *Setup* algorithm to generate system parameters. User A uses $KeyGen$ algorithm to obtain its secrete key tuple (PK_A, SK_A) and master verification key VK_A. Then User A uses $ShareKeyGen$ algorithm to generate m shared keys $\{SK_{A,i}\}_{i=1}^{m}$ for m key servers.

S1-1: *Setup* $\left(1^\lambda\right)$, input security parameter 1^λ and output system parameters:

$$\pi = (p, G, G_T, \hat{e}, g, F, H_\delta, H_\tau) . \tag{4}$$

Where, p is the prime number of λ digits, G and G_T are the prime number cyclic groups of p orders, $\hat{e}: G \times G \to G_T$ is bilinear map, g is the generator of G, $F: Z_p \times \{0,1\}^* \to Z_p$ is pseudo-random function, $H_\delta: Z_p \times \{0,1\}^* \to G_T$ and $H_\tau: \{0,1\}^* \to G$ are bilinear security hash functions.

S1-2: $KeyGen\,(\pi)$ input its system parameter π and outputs the key tuple (PK_A, SK_A, VK_A) of User A. $KeyGen$ function selects $x_1, x_2, x_3, x_4 \in_R Z_p$ and sets $PK_A = g^{x_1}$, $SK_A = x_1$, $VK_A = (x_2, x_3, x_4)$.

S1-3: $ShareKeyGen\,(\pi, SK_A, t, m)$ inputs system parameter π, secret key SK_A, threshold value t and m key servers and output m shared keys. $ShareKeyGen$ selects polynomial:

$$f_A(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \cdots + a_1 x + SK_A\,(\bmod p)\,. \tag{5}$$

Step 2 (storage stage): User A endows file M with identifier I_M and generates verification key $VK_A^{I_M}$ based on $VeriKeyGen$ algorithm. After that, User A decomposes file M into k file blocks $(m_1, m_2, \cdots, m_k) \in G_T^k$, encrypts file block m_i into ciphertext c_i based on Enc algorithm, and generates integrity label σ_i based on $TagGen$ algorithm. User A randomly matches key-label tuple (c_i, σ_i) for v storage server. Each storage server SS_j receives tuple and stores it as coding tuple $\left(c'_j, \sigma'_j\right)$ based on hybrid algorithm.

S2-1: $VeriKeyGen\,(\pi, VK_A, I_M, k)$ inputs system parameter π, master verification key VK_A, identifier I_M and file block quantity m and outputs the verification key $VK_A^{I_M}$ of M. Assumed that $VK_A = (x_2, x_3, x_4)$, it can be calculated that:

$$\begin{aligned} VK_A^{I_M} &= (F(x_2, I_M), F(x_3, I_M), H_\delta(x_3, I_M \|1)) \\ &= H_\delta(x_4, I_M \|2), \cdots, H_\delta(x_4, I_M \|k)\,. \end{aligned} \tag{6}$$

S2-2: $Enc\,(\pi, PK_A, I_M, m_1, \cdots, m_k)$ inputs system parameter π, key PK_A, identifier I_M and k file blocks $(m_1, \cdots, m_k)$, and outputs k groups of ciphertexts $(c_1, \cdots, c_k)$. Assumed that $PK_A = g^{x_1}$, it can be calculated that:

$$c_i = (g^{r_i}, \tau_M, m_i \hat{e}(g^{x_1}, \tau_M^{r_i}), \xi_i)\,. \tag{7}$$

Where, $r_i \in_R Z_p, \tau_M = H_\tau(l_M)$, $\xi_i \in Z_p^k$ is the unit vector of coordinate of 1.

S2-3: $TagGen(\pi, VK_A^{I_M}, c_i)$ inputs system parameter π, verification key $VK_A^{I_M}$ and ciphertext c_i and outputs integrity label σ_i. Assumed that $VK_A^{I_M} = (\delta, \delta', \delta_1, \delta_2, \cdots, \delta_k)$ and $c_i = (\alpha_i, \tau_M, \gamma_i, (g_{i,1}, g_{i,2}, \cdots, g_{i,k}))$, it can be calculated that:

$$\sigma_i = \hat{e}(\alpha_i, \tau_M)^\delta \cdot \gamma_i^{\delta/} \cdot \prod_{j=1}^{k} \delta_j^{g_{i,j}}\,. \tag{8}$$

Assumed

$$VK_A^{I_M} = (F(x_2, I_M), F(x_3, I_M), H_\delta(x_4, I_M||1),$$

$$H_\delta(x_4, I_M||2) \cdots, H_\delta(x_4, I_M||k))\,. \tag{9}$$

$$c_i = (g^{r_i}, \tau_M, m_i \hat{e}(g^{x_l}, \tau_M^{r_i}), \hat{\xi}_i)\,. \tag{10}$$

It can be g that:

$$\sigma_i = \hat{e}(g^{r_i}, \tau_M)^{F(x2,l_M)} \cdot (m_i \hat{e}(g^{x_1}, \tau_M^{T_i}))^{F(x_3,l_M)} \cdot H_\delta(x_4, I_M||i)\,. \tag{11}$$

S2-4: $Combine(\pi, (c_1, \sigma_1), \cdots, (c_k, \sigma_k), \boldsymbol{G}_j)$ inputs parameter π and k ciphertext label tuples $((c_1, \sigma_1), (c_2, \sigma_2), \cdots, (c_k, \sigma_k))$ and storage server combination coefficient $\boldsymbol{G}_j = (g_{j,1}, g_{j,2}, \cdots, g_{j,k})$, of which if SS_j does not receive (c_i, σ_i), then $g_{j,i} \in_R Z_p$ and $g_{j,i} \in_R Z_p$. Its output coding tuple is:

$$(c'_j, \sigma'_j) = (\odot_{i=1}^{k} c_i^{g_{j,i}}, \prod_{i=1}^{k} \sigma_i^{g_{j,i}})\,. \tag{12}$$

For SS_j, operation $\odot$ can be defined as follows: for the provided ciphertext $c_i = (g^{r_i}, \tau_M, m_i \hat{e}(g^{x_1}, \tau_M^{r_i})$, file block m_i and ciphertext $(g_{j,1}, g_{j,2}, \cdots, g_{j,k})$ as well as $(g_{j,1}, g_{j,2}, \cdots, g_{j,k})$, the ciphertext $c' = c_i^{\beta_i} \odot c_j^{\beta_j}$ of $m' = m_i^{\beta_i} \cdot m_j^{\beta_j}$ can be calculated under the circumstance of no key x_1 and block (m_i, m_j), where:

$$\begin{aligned}
&c_i^{\beta_i} \odot c_j^{\beta_j} \\
&= (g^{\beta_i r_i}, \tau_M, m_i^{\beta_i} e(g^{x_l}, \tau_M^{\beta_i r_i}), (\beta_i g_{i,1}, \cdots, \beta_i g_{i,k})) \\
&\odot (g^{\beta_j r_j}, \tau_M, m_j^{\beta_j} e(g^{x_l}, \tau_M^{\beta_j r_j}), (\beta_j g_{j,1}, \cdots, \beta_j g_{j,k})) \\
&= (g^{\beta_i r_i + \beta_j r_j}, \tau_M, m_i^{\beta_i} m_j^{\beta_j} e(g^{x_l}, \tau_M^{\beta_i r_i + \beta_j r_j}), \\
&(\beta_i g_{i,l} + \beta_j g_{j,1}, \cdots, \beta_i g_{i,k} + \beta_j g_{j,k}))\,.
\end{aligned} \tag{13}$$

Similarly, integrity label can be provided:

$$\sigma_i = \hat{e}(g^{r_i}, \tau_M)^{F(x2,I_M)} \cdot (m_i \hat{e}(g^{x_1}, \tau_M^{r_i}))^{F(x3,I_M)} \cdot \prod_{l=1}^{k} eH_\delta(x_4, I_M||l)^{gi,t}\,. \tag{14}$$

For c_i, the integrity label is:

$$\sigma_j = \hat{e}(g^{r_j}, \tau_M)^{F(x2,I_M)} \cdot (m_j \hat{e}(g^{x_1}, \tau_M^{r_j}))^{F(x_3,I_M)} \cdot \prod_{e=1}^{k} eH_\delta(x_4, I_M||l)^{g_j,t}\,. \tag{15}$$

For c_j, if verification key $VK_A^{I_M}$ and ciphertext (c_i, c_j) are unknown, calculate the integrity label $\sigma' = \sigma_i^{\beta_i} \cdot \sigma_j^{\beta_j}$ of $c' = c_i^{\beta_i} \odot C_j^{\beta_j}$, where:

$$\begin{aligned}
&\sigma_i^{\beta_i} \cdot \sigma_j^{\beta_j} \\
&= \hat{e}(g^{\beta_i r_i + \beta_j r_j}, \tau_M)^{F(x2,I_M)} \cdot (m_i^{\beta_i} m_j^{\beta_j} \hat{e}(g^{x_1}, \tau_M^{\beta_i r_i + \beta_j r_j}))^{F(x3,I_M)} \\
&\cdot \prod_{l=1}^{k} H_\delta(x_4, I_M\,||l)^{\beta_{i,g_{i,l}} + \beta_{j,j,l}} = TagGen(VK_A^{I_M}, c_i^{\beta_i} \odot c_j^{\beta_j})
\end{aligned} \tag{16}$$

Step 3 (integrity check stage): User A can send M integrity check request to key server KS_θ and then KS_θ inquires ω storage servers of ω coding tuples ω of I_M, where $\omega > k$. KS_θ combines coding tuples based on *Combine* algorithm to get $(c'_\theta, \sigma'_\theta)$ and uses $Verify$ algorithm to verify its integrity. If $(c'_\theta, \sigma'_\theta)$ is effective, KS conducts non-intersect group division on $((c'_{\theta_i}, \sigma'_{\theta_i})\}_{i=1}^{w}$, and verifies in recursion until the damaged tuple is found out. KS_θ informs User A of which storage service fails or suffers from damage; assumed that there are r such servers in total, then as for the small enough positive value Δ, if the grade r/ω is lower than the ineffectiveness rate $\varepsilon_0 - \Delta$, above integrity check process will be deemed effective.

If the integrity check fails, KS_θ is kept in query status and KS_θ algorithm is used until k groups of correct and linear independent tuples are found out. Besides, these correct tuple recovery systems can be used in recovery plan. If KS_θ can not find k groups of correct and linear independent tuples, extra background servers are required for system recovery.

S3-1: $Verify(\pi, VK_A^{I_M}, C'_\theta, \sigma'_\theta)$ inputs system parameter π, verification key $VK_A^{I_M}$ and combined tuple $VK_A^{I_M}$ and outputs the verification results. Assumed that:

$$VK_A^{I_M} = (\delta, \delta', \delta_1, \delta_2, \cdots, \delta_k)\,. \tag{17}$$

$$c'_\theta = (\alpha', \tau_M, \gamma', (g'_1, g'_2, \cdots, g'_k))\,. \tag{18}$$

The output will be effective if and only if:

$$\sigma'_\theta = \hat{e}(\alpha', \tau_M)^{\delta} \cdot \gamma'^{\delta'} \cdot \prod_{i=1}^{k} \delta_i^{g'_i}\,. \tag{19}$$

S3-2: *CheckRank*(π, κ) inputs system parameter π, correct tuple $\{(c'_{\theta_i}, \sigma'_{\theta_i})\}$ and combination coefficient matrix $\kappa = \left[g_{\theta_{i,j}}\right]$, and the output will be effective if and only if it satisfies in limited domain that:

$$Rank(\kappa) = k\,. \tag{20}$$

Step 4 (retrieval stage): User A sends retrieval request to m key servers and each key server sends request to u storage servers for u coding tuples $((c'_{\theta_i}, \sigma'_{\theta_i})\}_{i=1}^{u}$, and checks tuple individual based on $Verify$ algorithm while ignoring damaged individuals. After that, KS_i uses *ShareDec* to generate decryption token ζ_i among remained ciphertext $\{c'_{i_j}\}$, and returns back to $(\zeta_i, \{c'_{i_j}\})$. User A collects these ciphertexts and uses *CheckRank* algorithm to check whether its coefficient matrix grade is k or not, after that, User A finds out k groups of linear independent ciphertexts. If there are t decryption tokens $\{\zeta_{l_i}\}_{i=1}^{t}$ and k groups of linear independent ciphertexts $\{c'_{l_j}\}_{j=1}^{k}$ returning back, User A can use *Decode* algorithm to reconstruct original k groups of data blocks.

S4-1: *ShareDec*$(\pi, SK_{A,i}, c'_{i_j})$ inputs system parameter π, key $SK_{A,i}$ and ciphertext c'_{i_j} and outputs decryption token ζ_i. Assumed that $SK_{A,i} = f_A(i)$ and

$c'_{i_j} = (\alpha'_{i_j}, \tau_M, \gamma'_{i_j}, (g_{i_j,1}, \cdots, g_{i_j,k}))$, it can be calculated that:

$$\zeta_i = \tau_M^{f_A(i)}. \tag{21}$$

S4-2: $Decode(\pi, (\zeta_{l_l}, \cdots, \zeta_{l_t}), (c'_{l_1}, \cdots, c'_{l_k}))$ inputs system parameter π, t decryption tokens $\{\zeta_{l_l}, \cdots, \zeta_{l_t}\}$ and k groups of linear independent ciphertext k and outputs the reconstructed original k groups of data blocks $(m_1, \cdots, m_k)$:

(1) Reconstruct $\tau_M^{SK_A}$ in $\{\zeta_{l_l}, \cdots, \zeta_{l_t}\}$ with Lagrange interpolation and the process is as follows [14]:

$$\tau_M^{SK_A} = \prod_{i=1}^{t} ((\tau_M^{f_A(l_i)})^{\prod\limits_{j=1, j\neq i}^{t} -l_j/l_i - l_j}). \tag{22}$$

(2) Use $\tau_M^{SK_A}$ to decrypt $(c'_{l_1}, \cdots, c'_{l_k})$, assumed that $c'_{l_i} = (\alpha'_{l_i}, \tau_M, \gamma'_{l_i}, (g_{l_i,1}, \cdots, g_{l_i k}))$, it can be calculated that:

$$\begin{cases} M'_{l_i} = (m'_{l_i}, (g_{l_i,1}, \cdots, g_{l_i,k})) \\ m'_{l_i} = \dfrac{\gamma'_{l_i}}{\hat{e}(\alpha'_{l_i}, \tau_M^{SK_A})} = \prod\limits_{j=1}^{k} m_j^{g_{l_i,j}}. \end{cases} \tag{23}$$

S4-3: $Decode\left(M'_{l_1}, \cdots, M'_{l_k}\right)$ reconstructs file blocks $(m_1, \cdots, m_k)$. Assumed that the matrix $\kappa^{-1} = [\hat{g}_{l_i,j}]_{1 \le i,j \le k}$ obtained by $\left(M'_{l_1}, \cdots, M'_{l_k}\right)$ due to $\kappa = [g_{l_i,j}]_{1 \le i,j \le k}$ is κ inverse matrix, it can be calculated that:

$$m_i = \prod_{j=1}^{k} \left(m'_{l_j}\right)^{\hat{g}_{l_i,j}}. \tag{24}$$

4. Algorithm analysis

4.1. Algorithm complexity analysis

(1) Complexity of calculation process is: $VeriKeyGen$ process complexity is $2PRF + kHash$, where PRF denotes the verification times of master key and Hash denotes the Hashtable dimension adopted; complexity of $TagGen$ process is $1Pair + (k+2)\,Exp + (k+1)Mul$; the complexity of $Combine$ process is $kExp + (k-1)\,mul$; complexity of $kExp + (k-1)\,mul$ process is $1Pair + (k+2)\,Exp + (k+1)Mul$, where $1Pair + (k+2)\,Exp + (k+1)Mul$ denotes the quantity of label pairs, Exp denotes the quantity of integrity check and Mul denotes the times of cyclic execution.

(2) Complexity of data storage process is: complexity of user operation process is $3\,|p| + |I_M|$, where $3\,|p| + |I_M|$ is identifier dimension and p is the operation times of users; complexity of storage server operation process is $n \cdot L_T$, where L_T denotes the complexity of one storage server operation and n denotes the storage times; complexity of key server operation process is $m \cdot (2\,|p| + k \cdot L_T)$.

(3) Complexity of communication process is: complexity of storage process is $A \to KS : m \cdot (2\,|p| + k \cdot L_T)$, $A \to SS : k \cdot v \cdot L_T$; complexity of integrity check is $SS \to KS : \omega \cdot L_T$; complexity of retrieval process is $SS \to KS : m \cdot u \cdot L_T$, where m denotes communication times and other parameters are the same defined as above.

4.2. Security and robustness analysis

Theory 1: assumed that there are k tuples, n storage servers and m key servers, where $k \geq 13$, $n = ak$ and $m > (k-1)/[(n-1)\log_n(n/k)]$. Tuples are distributed among v storage servers randomly and its combination coefficient is the random number of v, where $v = b \ln k$ and $b > 5a$. If error and missing tuple accounts for $\varepsilon \leq n^{-(k-1)/[m(n-1)]} - k/n$ and each key server queries $u \geq (k-1)/[m \log_n(n/(\varepsilon n + k))]$ tuple, then the successful retrieval probability will be $1 - k/p - o(1)$ at least.

Demonstration: given that E_1 is the event of key server query less than the correct tuples and E_2 is the irreversible event of $k \times k$ matrixes constructed by k correct tuples, the successful retrieval probability will be $1 - pr\{E_1\} - pr\{E_2 \,|\overline{E_1}\} \cdot pr\{\overline{E_1}\}$, besides, there will be:

$$pr\{E_1\} = \binom{n - \varepsilon n}{k-1} \left(\frac{\varepsilon n + k - 1}{n}\right)^{um} = o(1)\ . \tag{25}$$

Then, when $u \geq (k-1)/\left[m1og_n(n/(\varepsilon n + k))\right]$, there is $n^{1-(k-1)/(um)} > \varepsilon n + k - 1$. To get the border of $pr\{E_2 \,|\overline{E_1}\} \cdot pr\{\overline{E_1}\}$, there is:

$$pr\{E_2 \,|\overline{E_1}\} \cdot pr\{\overline{E_1}\} \leq pr\{E_2 \,|\overline{E_1}\} \leq \frac{k}{p} + o(1)\,. \tag{26}$$

So the successful retrieval probability is:

$$\begin{aligned} &1 - pr\{E_1\} - pr\{E_2 \,|\overline{E_1}\} \cdot pr\{\overline{E_1}\} \\ &\geq 1 - o(1) - \frac{k}{p} - o(1) = 1 - \frac{k}{p} - o(1) \end{aligned} \tag{27}$$

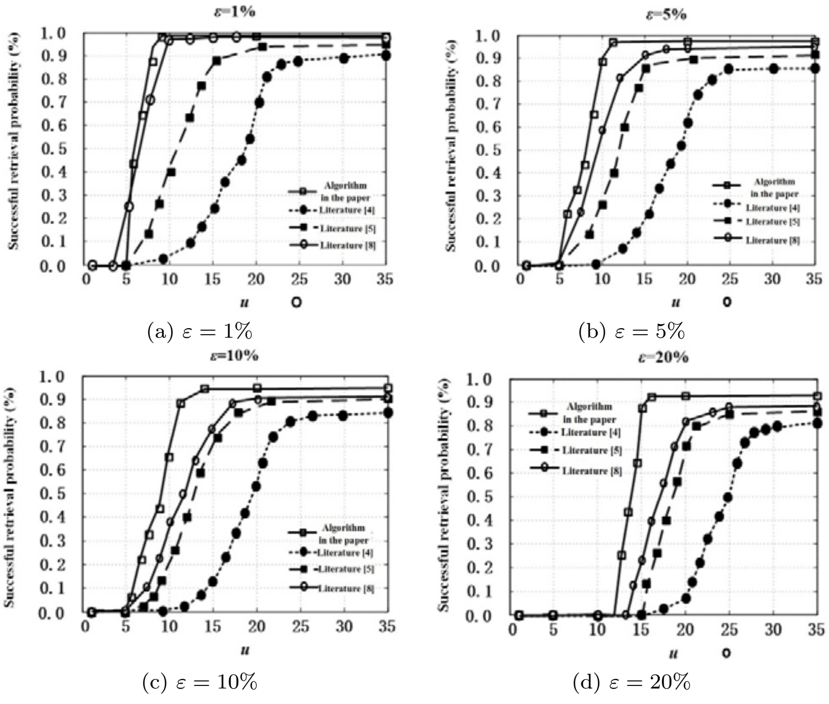
Demonstration completed.

4.3. Experimental analysis

The theory 1 has built the lower limit of parameter setting way and successful retrieval probability aimed at storage server damage and failure. To prove the robustness of parameter defined data determined in integrity check program, set analog system $n = 1024$, $k = 256$, $m = 64$ and $v = 111$ and test $u = 1 \sim 35$ to get the successful retrieval probability of data, where p is the prime number with 513 digits. The test aims to prove the quantity of storage servers queried by key servers, which is much lower than the theoretical value. Compare algorithms by selecting literature [4], literature [5] and literature [8] and select data damage rate of $\varepsilon = 1\%$, 5%, 10% and 20%; the comparison of these successful retrieval probabilities are shown in Fig.

3.

Fig. 3 shows the comparison of the successful retrieval probabilities of the algorithm of the paper and that in literature [4-5] and literature [8]. Seen from the figure, the quantities of query servers required for several comparison algorithms under the same successful retrieval probabilities increase with the increase of data damage rate, of which the quantity of query servers required for the algorithm of the Paper is less than that of comparison algorithms and its index of successful retrieval probability is also higher than that of comparison algorithms, similarly, above index of algorithm in literature [4] is superior than that in literature [5]. Under the circumstance of relatively small data damage rate, algorithm in literature [8] has better index of successful retrieval probability, which is closer to the algorithm in the Paper; however, along with the increase of data damage rate, the reduction amplitude of algorithm in literature [8] is larger than that of the algorithm in the Paper and approaches to the algorithm in literature [5] gradually.



(a) $\varepsilon = 1\%$ (b) $\varepsilon = 5\%$

(c) $\varepsilon = 10\%$ (d) $\varepsilon = 20\%$

Fig. 3. Probability of successful retrieval

Table 1 provides the comparison of computing time and successful retrieval probability under the circumstance that the algorithm in the Paper, literature [4, 5] and literature [8] selects data damage rate $\varepsilon = 1\%$, 5%, 10% and 20% respectively.

Table 1. Probability of successful retrieval

Algorithm	Index	1%	5%	10%	20%
Algorithm in the Paper	Successful retrieval probability (%)	99.5	98.1	95.2	91.4
	Computing time (s)	5.3	7.6	9.8	11.4
Algorithm in literature [4]	Successful retrieval probability (%)	95.5	91.2	90.3	86.7
	Computing time (s)	8.6	9.7	11.5	14.8
Algorithm in literature [5]	Successful retrieval probability (%)	91.3	86.7	83.2	80.8
	Computing time (s)	9.6	10.7	12.5	16.1
Algorithm in literature [8]	Successful retrieval probability (%)	98.3	94.3	91.9	88.4
	Computing time (s)	5.6	8.5	10.6	12.8

Seen from the data in Table 1, under the circumstance of $\varepsilon = 1\%$, 5%, 10% and 20%, the successful retrieval probability indexes of algorithm in the Paper are 99.5%, 98.1%, 95.2% and 91.4% respectively, higher than the comparison algorithms in literature [4-5] and literature [8], showing a relatively higher robustness of retrieval quality and data check under data damage. Under the circumstance of data damage rate selects $\varepsilon = 1\%$, 5%, 10% and 20%, the computing time indexes of algorithm in the Paper are 5.3s, 7.6s, 9.8s and 11.4s respectively, faster than the comparison algorithms in literature [4-5] and literature [8], of which the computing time of algorithm in literature [8] is the most closest to that of the algorithm in the Paper. Although the algorithm in the Paper requires increasing calculation complexity on the part for integrity check, the quantities of query servers required are relatively reduced, making the algorithm save a certain query time and showing the relatively high computational efficiency.

5. Conclusion

In the Paper, a secure erasure code storage system considering both data confidentiality and data robustness is proposed and pseudo-random bilinear map is used to construct the threat model of integrity check strategy of cloud storage, realizing the supplement of secure erasure code storage system algorithm function of multiple key servers. The simulation result shows the algorithm has a relatively high computational efficiency and high successful retrieval probability, showing the relatively good performance of the algorithm. In the future research, efforts shall be concentrated on the improvement of parameters setup and the demonstration of effectiveness theory of system prototype and proper application of the theory.

References

[1] Kosmas P, Miao Z (2015) *Novel inversion tools to improve performance of the DBIM algorithm for microwave medical imaging*[C]// IEEE Mtt-S International Microwave Workshop Series on Rf and Wireless Technologies for Biomedical and Healthcare Ap-

plications. IEEE, 2015:1-3.

[2] Kosmas P, Miao Z (2015) *Novel inversion tools to improve performance of the DBIM algorithm for microwave medical imaging*[C]// IEEE Mtt-S International Microwave Workshop Series on Rf and Wireless Technologies for Biomedical and Healthcare Applications. IEEE, 2015:1-3.

[3] Kosmas P, Miao Z (2015) N*ovel inversion tools to improve performance of the DBIM algorithm for microwave medical imaging*[C]// IEEE Mtt-S International Microwave Workshop Series on Rf and Wireless Technologies for Biomedical and Healthcare Applications. IEEE, 2015:1-3.

[4] Bharghavan V, Lee K W, Lu S, et al. (1998) *The TIMELY adaptive resource management architecture*[J]. Personal Communications IEEE, 5(4):20-31.

[5] Grilo A, Jardim-Goncalves R (2010) *Value proposition on interoperability of BIM and collaborative working environments*[J]. Automation in Construction, 19(5):522-530.

[6] Gulisano V, Jiménezperis R, Soriente C, et al. (2012) *StreamCloud: An Elastic and Scalable Data Streaming System*[J]. IEEE Transactions on Parallel & Distributed Systems, 23(12):2351-2365.

[7] Kaneko Y, Lapusta N, Ampuero J (2008) *Spectral element modeling of spontaneous earthquake rupture on rate and state faults: Effect of velocity-strengthening friction at shallow depths*[J]. Journal of Geophysical Research Solid Earth, 113(B9):269-283.

[8] Tronconi E, Groppi G (2000) *A study on the thermal behavior of structured plate-type catalysts with metallic supports for gas/solid exothermic reactions*[J]. Chemical Engineering Science, 55(24):6021-6036.

[9] Nepal M P, Staub-French S, Pottinger R, et al. (2012) *Querying a building information model for construction-specific spatial information*[J]. Advanced Engineering Informatics, 26(4):904-923.

[10] Wang X, Shilman M, Raghupathy S (2006) *Parsing ink annotations on heterogeneous documents*[C]// Eurographics Conference on Sketch-Based Interfaces and Modeling. Eurographics Association, 2006:43-50.

[11] Kaneko Y, Ampuero J P, Lapusta N (2011) *Spectral-element simulations of long-term fault slip: Effect of low-rigidity layers on earthquake-cycle dynamics*[J]. Journal of Geophysical Research Solid Earth, 116(B10):1241-1249.

[12] Elbim C, Hakim J, Gougerot-Pocidalo M A (1998) *Heterogeneity in Lewis-X and sialyl-Lewis-X antigen expression on monocytes in whole blood: relation to stimulus-induced oxidative burst.*[J]. American Journal of Pathology, 152(4):1081-90.

Received May 7, 2017