Research on Data Auditing Technology Based on PARCC

Jindong He1,*, Zhijun Tang1, Zhichao Zhao1, Fucai Luo2, Chenghua Lin1 and Dan Wu1

1State Grid Fujian Electric Power Research Institute, Fuzhou Fujian 350007, China
2State Grid Fujian Electric Power Company Limited, Fuzhou Fujian 350003, China

*Corresponding author e-mail: 2468537873@qq.com

Abstract. In order to improve the data security in the computing environment, a "public auditing" scheme PARCC was proposed for the regenerative coding data in the cloud computing system. PARCC allows the user to use a third-party auditor TPA to remote check the integrity of the data. In addition, in order to solve the data owner without online failure data/signature of regeneration, was introduced to the system in one and a half-trusted agent, and designed a new data signature scheme, allowing the agent to replace the data owner to provide renewable data valid signatures. Considering the privacy protection of data, the random cover method of coefficient in PARCC scheme ensures that the user data will not be leaked during the audit and repair process. The safety of the scheme is proved, and the performance of the scheme is comprehensively evaluated and analyzed through the theory. The results show that the scheme has good performance.

1. Introduction

In the cloud service mode, the correctness, reliability and integrity of data are faced with certain security risks due to data stored on remote cloud servers, thus limiting its development and application in practice [1-2].

In the actual cloud computing production environment, user data is faced with security threats from multiple aspects. On the one hand, there are a large number of attackers from inside and outside, who will actively delete or damage user data. On the other hand, when cloud service providers are faced with data loss or destruction, they may deliberately hide these details for reasons such as maintaining their own reputation or capital, and claim that the data is still correct, resulting in the user data no longer available [3-5]. Therefore, it is of great practical significance and theoretical value to design an efficient data audit protocol to conduct periodic verification of user data in cloud computing system to ensure the correctness and integrity of data in cloud storage.

At present, the academic community has done a lot of research on the data integrity verification technology, and proposed a variety of data audit schemes. Literature [6] propose a novel privacy-preserving mechanism that supports public auditing on shared data stored in the cloud. We exploit ring signatures to compute verification metadata needed to audit the correctness of shared data, and the experimental results demonstrate the effectiveness and efficiency of our mechanism when auditing shared data integrity [6]. Literature [7] present a novel public auditing scheme named MUR-DPA. Compared to existing integrity verification and public auditing schemes, theoretical analysis and experimental results show that the proposed MUR-DPA scheme can not only incur much less
communication overhead for both update verification and integrity verification of cloud datasets with multiple replicas, but also provide enhanced security against dishonest cloud service providers [7]. Literature [8] provide a formal analysis for possible types of fine-grained data updates and propose a scheme that can fully support authorized auditing and fine-grained update requests, and based on our scheme, we also propose an enhancement that can dramatically reduce communication overheads for verifying small updates [8].

This paper proposes a cloud computing data audit scheme to improve the security of the data transmission process.

2. Preliminary knowledge

2.1. Random network based regeneration code

Regeneration code construction method based on random network coding:

Given a file F contains m raw blocks \( \{ \overline{w}_i \}_{i=1}^{m} \), suppose each of these blocks contains s symbols \( \{ \overline{w}_i \} = (\overline{w}_{i_1}, \overline{w}_{i_2}, ..., \overline{w}_{i_m}) \). Each data symbol \( \overline{w}_{ij} \) is contained in the finite field \( GF(2^w) \). The data owners encode these data blocks. The specific calculation process of each code block is as follows: m random coding coefficients \( \{ \epsilon_i \}_{i=1}^{m} \) are selected from the finite field \( GF(2^w) \), and the coding block is the linear combination \( \overline{w} = \sum_{i=1}^{m} \epsilon_i \cdot \overline{w}_i \) of the original data block. When \( GF(2^w) \) is large enough, any m of the above coding blocks are linearly independent with a very high probability. Therefore, the user can obtain m original data blocks by solving a set of m element equations, and then recover the original data file. Suppose all the coded data blocks are uniformly distributed to n storage servers. \( \alpha \) blocks are stored on each server. When \( k \alpha \geq m \), the user can use the data sharing on any k servers to reconstruct the original file (that is, the system satisfies the MDS attribute). When a failed server needs to be repaired, the repair process connects with l health servers and downloads \( \beta \) blocks from each server. Using these blocks, the process can complete the repair of the failed data to guarantee the MDS properties of the system. FIG. 1 shows an example diagram of regenerative code based on random network coding, and its parameter selection references [10].

![Figure 1](image)

Figure 1. Shows the regeneration code based on random network coding

2.2. Linear subspace derived from regenerative code

A linear subspace with dimension m can be derived by using data file F. The method is as follows: Given that the file F is divided into m data blocks, each data block contains s symbols, each block can be regarded as an s-dimensional vector \( \overline{w}_i \in GF(p)^s \). before the coding operation, we extend each vector in the following way:

\[
\overline{w}_i = (\overline{w}_{i_1}, \overline{w}_{i_2}, ..., \overline{w}_i, \underbrace{0, ..., 1, 0, ..., 0}_{m-i}) \in GF(p)^{s \times m} \tag{1}
\]
The original ith data block vector \( \bar{w}_i \) was appended with a unit vector of length \( m \), and the ith position of the unit vector was 1, and the rest was 0.

The extended original data is encoded into \( n \) blocks. Select the random coefficient \( \varepsilon_i \in GF(p) \) in the finite domain, and each code block is a linear combination of these original vectors:

\[
v = \sum_{i=1}^{m} \varepsilon_i w_i \in GF(p)^{r+m}
\]  

(2)

The latter half \( m \) elements in vector \( v \) are the values of the coding coefficient \( \varepsilon \) of this data block:

\[
v = (v_{s1}, v_{s2}, \ldots, v_{s\alpha}, \varepsilon_1, \ldots, \varepsilon_m) \in GP(p)^{r+m}
\]  

(3)

Thus, an \( m \)-dimensional linear subspace \( V \) is constructed, and the extended vector set \( \{w_1, w_2, \ldots, w_m\} \) is a basis for this subspace.

Constructing linear subspace \( V \) based on regenerative code can quickly generate signatures for all coding data blocks in the cloud computing system.

With this efficient signature generation method, the user only needs to perform the signature operation for \( m \) basis vectors at the initial stage, which effectively improves the efficiency of signature set generation.

2.3. Bilinear pairwise mapping

Let \( G \) and \( G_T \) be two \( p \) factorial cyclic groups. If the mapping \( e: G \times G \rightarrow G_T \) meets the following conditions, it is called bilinear pairwise mapping:

Bilinear: For any \( u, v \in G \) and \( a, b \in \mathbb{Z}_p^* \), it satisfies \( e(u^a, v^b) = e(u,v)^{ab} \), and this property can be extended to multiplication, that is, for any \( u_1, u_2, v \in G \), it satisfies \( e(u_1 \cdot u_2, v) = e(u_1, v) \cdot e(u_2, v) \);

Non-degeneracy: If \( g \) is the generating element of group \( g \), \( e(g, g) \) is also the generating element of \( G_T \);

Computability: For any \( u, v \) for \( G \), there are efficient algorithms for calculating the value of \( e(u, v) \) in polynomial time.

The public audit model of regenerative code data in cloud computing system is shown in figure 2, mainly including data owner, cloud server, third-party auditor (TPA) and agent.

![Figure 2](image_url)

Figure 2. Public audit model of regenerative code data in cloud computing system
The security threats to the model shown in figure 2 mainly come from three aspects: the end server, the third-party auditor TPA, and the semi-trusted agent.

In order to ensure the correct and efficient verification of data integrity and protect the confidentiality and availability of user data, the following requirements need to be considered in the design of the scheme: support for public verification, complete protocol, privacy protection, signature repair, and wrong positioning. The Audit scheme mainly includes three processes: Setup, Audit and Repair.

3. PARCC scheme design
The working parameter of the regenerative code in the cloud computing system is \((n, k, l, \alpha, \beta)\). For simplicity, it is assumed that \(\beta = 1\). Let \(G\) and \(GT\) be the multiplication cyclic group of two order \(p, e:\ G \times G \rightarrow G_T\) bilinear pair mapping, \(g\) is the generating element of group \(G\). \(H(\cdot) : \{0,1\}^* \rightarrow G\) is the safe hash function that maps any length bit string to elements in group \(G\). The main symbols and terms used in the scenario description are listed in table 1.

| Symbol | Meaning |
|--------|---------|
| \(m\)  | number of raw data blocks |
| \(s\)  | the number of data segments in the original data block |
| \(w_k\) | the \(k\)th data segment of the original data block \(w_i\) |
| \(v_{ij}\) | the \(j\)th code block on server \(i\) |
| \(v_{ijk}\) | the \(k\)th data segment of the coding block \(v_{ij}\) |
| \(\epsilon_{ij}\) | the \(\lambda\) coefficient of the coding block \(v_{ij}\) |
| \(t\)  | file identifier |
| \(\sigma_{ijk}\) | signature of data segment \(v_{ijk}\) |
| \(\Phi_i\) | the signature collection of all data blocks on server \(i\) |
| \(\psi_i\) | set of all coded data blocks on server \(i\) |
| \(C\)  | challenge values in the data audit process |
| \(p\)  | evidence in data audits, include aggregate data segment \(\mu_i\), polymerase signature \(\sigma_i\), \(\{p_{i1}, p_{i2}, ..., p_{im}\}\) - auxiliary information for coefficient integrity check |
| \(C_r\) | data recovery statement |
| BA     | cloud server response during data recovery |

Setup: This process initializes the relevant parameters in the audit scheme, and generates the regeneration code and signature.

KeyGen \((1^K) \rightarrow (pk, sk)\): The data owner generates a public-private key pair \((spk, ssk)\) for the standard signature, selects two random elements \(x, y \leftarrow \mathbb{Z}_p\), and computes \(pk_x \leftarrow g^x, pk_y \leftarrow g^y\). After that, the data owner keeps \(sk = (x, y, ssk)\) as the secret parameter safely locally and releases \(pk = (pk_x, pk_y, spk)\) publicly.

Delegation \((sk) \rightarrow (x)\): The data owner users the agent's public key to encrypt \(x\) and send it to the agent. After receiving the secret message, the agent performs the private key decryption operation and saves the value of \(x\) locally.
SigAndBlockGen (sk, F) → (Φ, Ψ, t): For data file F, the data owner selects a random identifier

\( ID \leftarrow \{0,1\}^* \). A random symbol \( u \leftarrow G \) and a set of random symbols \( \Gamma = \{w_1, w_2, \ldots, w_m \} \). The elements in this collection \( w_i \leftarrow G \). Tag of data owner generated file F:

\[
t = (ID||u||w_1||\ldots||w_m) \ || \text{Sig}_{\text{sk}}(ID||u||w_1||\ldots||w_m)
\]

(4)

Among them, \( \text{Sig} () \) represents a standard digital signature scheme. The scheme assumes that the original file F is divided into \( m \) data blocks \( \{w_i\}_{i=1}^m \), and each block is divided into \( s \) data blocks. All \( nm \) coding blocks after coding are distributed evenly to \( n \) cloud servers.

Data block extension: the data owner extends the original \( m \) blocks in the format of formula (1).

Original data block signature: the data owner takes the data part of the extended data block as a collection of data segments and computes the signature for each segment as follows:

\[
\sigma_{j,k}^{*} = (u^m v_k \prod_{\lambda=1}^{m} w_{\lambda}^{\alpha_{(\lambda)}^{(\lambda)}})^{v} = (u^m v_k \cdot w_{\lambda})^{v}
\]

(5)

Where \( 1 \leq j \leq m, 1 \leq k \leq s \).

Data encoding and signature aggregation: The data owner randomly selected \( m \) elements \( \{e_{j,k}^{(\lambda)}\}_{\lambda=1}^{m} \) from \( GF(p) \) as the coding coefficient, and then the linear combination of the original data block was used to get the coded data block \( v_j^{(1 \leq i \leq n, 1 \leq j \leq \alpha)} \).

\[
v_j = \sum_{\lambda=1}^{m} e_{j,k}^{(\lambda)} w_{\lambda} \in GF(p)^{v:m}
\]

(6)

Obviously, each symbol \( v_{j,k} \) of the coding block can be represented as:

\[
v_{j,k} = \sum_{\lambda=1}^{m} e_{j,k}^{(\lambda)} w_{\lambda} \in GF(p)
\]

(7)

Where \( 1 \leq k \leq s + m \).

The original data block signature is aggregated by using formula (8):

\[
\sigma_{j,k}^{*} = \prod_{\lambda=1}^{m} \sigma_{j,k}^{*} = (u^m v_k \prod_{\lambda=1}^{m} w_{\lambda}^{\alpha_{(\lambda)}^{(\lambda)}})^{v}
\]

(8)

Where \( 1 \leq k \leq s \).

Signature generation: the data owner generates the signature for each segment \( v_{j,k} \).

\[
\sigma_{j,k} = H(ID \ || \ j \ || \ k \ || \ x) \cdot \sigma_{j,k}^{*} = H(ID \ || \ j \ || \ k \ || \ x) (u^m \prod_{\lambda=1}^{m} w_{\lambda}^{\alpha_{(\lambda)}^{(\lambda)}})
\]

(9)

Where \( i \in [n], j \in [\alpha], k \in [s] \) respectively represent the server index value in the cloud computing system, the data block index value on a server and the data segment index value in a coding block.

In SigAndBlockGen (sk, F) the last stage of the algorithm, got the coding block set \( \Psi = \{\psi_i = \{v_j\}_{j \leq s} \}_{i \leq n} \) and signature collection \( \Phi = \{\Phi_i = \{\sigma_{j,k}\}_{j \leq s} \}_{i \leq n} \). The data owner
distributes elements from both sets to n cloud servers and deletes local copies. Specific, data owner will \( \{\Psi, \Phi, I, I, t\} \) the first i sent to a server.

Audit: Third-party auditor TPA audits data on all n servers by random sampling to verify that each server holds \( \alpha \) code block correctly. In addition, in order to effectively reduce the network bandwidth overhead, we conducted a batch audit on each server.

Challenge (Finfo) \( \rightarrow \) (C): Taking the file information Finfo as input, TPA is a set \( Q_i = \{(k^*_x, \alpha^*_z)\}_{1 \leq z \leq c} \) containing c elements for each server I (1 \( \leq i \leq n \)), where \( k^* \in_R [1, 8] \) is the index value of the data section sampled randomly and \( \alpha^* \leftarrow GF(p) \) is the random number corresponding to it. In addition, in order to implement batch audit, TPA also needs to generate another random number set \( \Lambda_i = \{\alpha^*_z\}_{1 \leq z \leq a} \), \( \alpha^*_z \leftarrow GF(p) \). Finally, TPA will challenge the value of \( C = \{Q_i, \Lambda_i\} \) sent to the server i.

ProofGen(C, \( \Phi, \Psi \)) \( \rightarrow \) (p): Server i to receive the challenge of TPA was sent value \( \{Q_i, \Lambda_i\} \), computing (10) in the first place.

\[
\mu_j = \sum_{r=1}^{c} \alpha^*_r \psi_{jk}^r, \sigma_{ij} = \prod_{r=1}^{c} \sigma_{jk}^r
\]

The \( j \in [1, \alpha] \), server I use \( \Lambda_i \) random numbers in the polymerization of \( \mu_{ij} \), and \( \sigma_{ij} \):

\[
\mu_i = \sum_{j=1}^{\alpha} \alpha_j \mu_{ij}, \sigma_{ij} = \prod_{j=1}^{\alpha} \sigma_{ij}
\]

In order to ensure that TPA can verify the correctness of the coding coefficient at the same time, server i calculates m values according to formula (12):

\[
p_{i, \lambda} = (\sum_{j=1}^{\alpha} \alpha_j \epsilon_{ij}) \cdot \sum_{r=1}^{c} \alpha^*_r
\]

Where \( 1 \leq \lambda \leq m \), \( \Theta_i = \{p_{ij}, p_{i2}, \ldots, p_{im}\} \).

Finally, server i sends aggregate evidence to TPA

Verify(P, pk, C) \( \rightarrow \) (0, 1): When the evidence \( p \) of server i response is received, TPA of the third party auditor first uses SPK to verify the digital signature in the file label t to ensure that the received response is indeed from the data file that needs to be audited. If validation fails to pass, the termination of the validation process, on the contrary, u and TPA extract random symbols \( \Gamma = \{w_1, w_2, \ldots, w_m\} \) and perform subsequent validation.

\[
RHS = e\left(\prod_{j=1}^{\alpha} \prod_{r=1}^{c} H_{\psi_{jk}^r}^{\alpha^*_r}, pk_i^x\right) \cdot e(u^x, pk_x)
\]

\[
e\left(\prod_{\lambda=1}^{m} w_{i, \lambda}^{-p_{i, \lambda}}, pk_y\right) \cdot e(\sigma_y, g) = RHS
\]

If the equation (14) is true, TPA output 1 indicates that the data integrity verification is passed; otherwise, output 0 indicates that the data is invalid.

Data Repair (Repair): Assume that TPA specifies the number \( \eta \) of the failed server.
ClaimForRep (Finfo) → (Cr): The agent randomly selects 1 health server \{i_0\}_{1 \leq i \leq s} and establishes a connection with it. For any server \(i \in \{i_0\}_{1 \leq i \leq s}\), proxy generates a set of random coefficients \(A_i = \{a_i\}_{1 \leq i \leq s}\), \(a_i \in GF(p)\). Next, the agent will repair statement \(C_i = \{A_i\}\) sent to the server \(i\).

GenForRep(Cr, \(\Phi\), \(\Psi\)) → (BA): Server \(i\) receive to repair the coefficient on the server on that Cr after using \(\Lambda_i\) stored locally linear combination coding block to generate a new help data block, at the same time generate an aggregate signature.

\[
\tilde{v}_i = \sum_{j=1}^{\alpha} \alpha_j v_j, \tilde{\sigma}_{jk} = \prod_{j=1}^{\alpha} \sigma_{jk}^i
\]

(15)

Where \((1 \leq k \leq s)\). Finally, server \(i\) replies to the proxy response set \(BA_i = \{\tilde{v}_i, \{\tilde{\sigma}_{jk}\}_{1 \leq k \leq s}\}\).

BlockAndSigReGen (Cr, BA) → (Φ′, Ψ′): Considering each block \(\tilde{v}_i\) as \((\tilde{v}_{i1}, \ldots, \tilde{v}_{i\alpha}, \tilde{\sigma}_{i1}, \ldots, \tilde{\sigma}_{i\alpha})\), the agent will verify whether the following \(k\) equations are true:

\[
e(\prod_{i \in \{i_0\}} \tilde{\sigma}_{jk}), g = e(\prod_{i \in \{i_0\}} H_{jk}^{i'}, pk_j) \cdot e(u \sum_{w \in \omega^i} \lambda w, pk_j) \cdot e(\prod_{i \in \{i_0\}} w \sum_{w \in \omega^i} \lambda w, pk_j)
\]

(16)

Where \((1 \leq k \leq s, H_{jk} = H(ID||i||j||k))\). Assuming that the repaired code block and its signature will be stored on a newly added server \(\eta'\), the proxy rebuilds the failed \(\alpha\) blocks by: Select the random number \(z_i \leftarrow GF(p), i \in \{i_0\}_{1 \leq i \leq s}\), fix failed data blocks:

\[
v_{j'k} = \sum_{i \in \{i_0\}} z_i \tilde{v}_i
\]

(17)

Generate valid signatures for each data segment:

\[
\sigma_{j'k} = T_{jk}^{x} \cdot \prod_{i \in \{i_0\}} (\tilde{\sigma}_{jk})^{z_i}
\]

(18)

Where \((1 \leq j \leq \alpha, 1 \leq k \leq s)\), the conversion factor \(T_{jk}\) is:

\[
T_{jk} = \frac{H(ID || \eta' || j || k)}{\prod_{i \in \{i_0\}} \prod_{j' = 1}^{\alpha} H(ID || i' || j' || k)}
\]

(19)

Finally, the agent sends the regenerative/restored collection \(\psi' = \{v_{j'k}\}_{1 \leq j \leq \alpha, 1 \leq k \leq s}\) and signature collection \(\phi' = \{\sigma^i_{j'k}\}_{1 \leq k \leq s, 1 \leq j \leq \alpha}\) to the server’s newly added server \(\eta'\). At this point, the data repair process is complete.

4. Safety analysis

4.1. Accuracy analysis

Verify the correctness of the PARCC scheme data Audit and data Repair respectively.

Theorem 1: Given a server \(i\), which have a block of data collection are \(\Psi_i\) and related signature collection \(\Phi_i\). Auditor TPA in data audit phase to the correct authentication server \(i\) to \(\Psi_i\) and \(\Phi_i\) hold, the agent in data repair phase can download the correct check is the integrity of the data block.
Proof (14) is correct
\[
e(\sigma_j, g_2) = e(\prod_{j=1}^{c} \sigma_{ji}^{g_j}, g_2) = e(\prod_{j=1}^{c} H_{jjr}^{\sigma_{ji} g_j}, pk_j) \cdot e(u^{u_j}, pk_j) \cdot e(\prod_{j=1}^{m} w_{j}^{pk_j}, pk_j)
\]  

(20)

Can prove that
\[
e(\prod_{j=1}^{m} w_{j}^{pk_j}, pk_j) \cdot e(\sigma_j, g_2) = RHS
\]  

(21)

Prove the correctness of formula (16):
\[
e(\prod_{i \in \{\mu\}} \tilde{\sigma}_{ik}, g_2) = e(\prod_{i \in \{\mu\}} H_{ik}^{\sigma_{ik} g_2}, pk_j) \cdot e(u^{\sum_{\omega(k)} w_{ik}}, pk_j) \cdot e(\prod_{j=1}^{m} w_{j}^{\sum_{\omega(k)} w_{ik}}, pk_j)
\]  

(22)

4.2. Regeneration cannot be used as modeling

Theorem 2: Unless the enemy (or agent) performs the Repair process correctly, the probability that it can generate a valid signature for an invalid code block and pass the next validation is negligible.

Proof: Any data blocks represented by non-zero vector \( v^* \in GF(p)^{e+m} \) that are not part of the basis vector \( \{w_j\}_{j=1}^{m} \) generating subspace V are invalid data. These invalid data may cause the original data to be unrecoverable after several data repair processes. Suppose each data segment contains \( \zeta \) symbol. The k-th data segment of a coding block \( v \) is regarded as a vector \( \bar{v}_k = (v_{k1}, v_{k2}, ..., v_{k\zeta}) \in GF(p)^{\bar{e}} \). After adding m relevant coding coefficients, the extended vector \( \bar{v}_k = (v_{k1}, v_{k2}, ..., v_{k\zeta}) \in GF(p)^{\bar{e}} \) can be obtained.

Let \( V_s \) be a subspace spanned by \( m \) basis vectors, and each basis vector is obtained by extending (by referring to formula 1) the k-th segment of the original data block. Any non-zero vector \( v_k \) only represents a valid data segment in a subspace \( V_s \), otherwise it is not valid. Referring to formula (9), the signature for the KTH data section can be obtained as follows:

\[
\sigma_k = H(ID || ...)^{v_k} (\prod_{r=1}^{\zeta} u_{kr}^{v_{kr}} \prod_{j=1}^{m} w_{j}^{v_{jk}})^{\bar{e}}
\]  

(23)

Considering that the agent has mastered the secret value \( x \), it is only necessary to prove that the signature described in the following formula cannot be forged without mastering the secret key \( y \):

\[
\bar{\sigma}_k = (\prod_{r=1}^{\zeta} u_{kr}^{v_{kr}} \prod_{j=1}^{m} w_{j}^{v_{jk}})^{\bar{e}}
\]  

(24)

Take each \( w_{j} \) as the output value of the independent hash function \( H_{\alpha}(ID || \lambda) \), and we can get:

\[
\bar{\sigma}_k = (\prod_{r=1}^{\zeta} u_{kr}^{v_{kr}} \prod_{j=1}^{m} H_{\alpha}(ID || \lambda)_{\zeta})^{\bar{e}}
\]  

(25)

Finally, we can conclude that the probability that the agent cannot generate valid verifiable signature \( \sigma_k \) for invalid segment \( v_k \) is negligible. It is computationally unfeasible for an adversary to generate valid signatures for a vector that is not a member of a subspace \( V_s \).
5. Performance comparison

5.1. Performance comparison
Table 2 shows the comparison between the PARCC scheme proposed in this paper and other remote verification mechanisms for regenerative code data [11-12]. \( s \) is the number of the data contained in each block, \( m \) is the data file for the number of the original data block segmentation, \( \alpha \) is each number of block of data stored on the server, and \( k' \) and \( n' \) are AECC (Adversary Error Correcting Code) yards of design parameters.

| Plan                        | The plan of B.Chen | The plan of H.Chen | PARCC |
|-----------------------------|--------------------|--------------------|-------|
| public audit support        | No                 | No                 | Yes   |
| support privacy             | Yes                | Yes                | Yes   |
| support user offline        | No                 | No                 | Yes   |
| failed server localization time | \( O(1) \)     | \( O(C_k^d(n-k)\alpha) \) | \( O(1) \) |
| server storage overhead     | \( (s\alpha + ma + \alpha)\mid p \) | \( (nm\alpha + s(n' + k')\alpha + \alpha)\mid p \) | \( (sa + ma)\mid p \) |
| communication overhead during audit | \( (m+2)\alpha \mid p \) | \( (nm + c)\alpha \mid p \) | \( (m + 2) \mid p \) |
| repair phase communication overhead | \( (ma + s + 1)\mid p \) | \( (nm\alpha + s + 1)\mid p \) | \( (2s + m)\mid p \) |

5.2. Results analysis
As can be seen from table 2, the scheme in literature [11-12] only supports private auditing, while the PARCC scheme allows third parties other than the data owners to remotely verify the data integrity of the cloud computing system, and at the same time realizes the privacy protection of users' original data. The PARCC solution avoids the data owner's data online maintenance burden, while both the b. Chen solution and h. Chen solution require the data owner to stay online all the time to cope with frequent failure data repair problems. The PARCC scheme and b. Chen scheme can successfully locate the failed server in the process of one audit, and the calculation cost is greatly reduced. In terms of cloud server storage cost, PARCC solution is slightly lower than b. Chen solution. In terms of communication cost, both PARCC and b. Chen solution adopt batch audit strategy, which is greatly optimized compared with H.Chen solution. B. Chen. The communication overhead in the audit of the scenario is A times that of PARCC.

6. Concludes
A public audit scheme PARCC is proposed for cloud computing system based on regenerative code, which allows the data owner to authorize trusted third-party auditor TPA to verify the integrity of remote data. In order to protect user data privacy, the PARCC scheme randomly covers the coding coefficient to ensure that TPA cannot reconstruct the original data. In order to further ease the data owner's data maintenance burden online, a semi-trusted agent is introduced into the schema system model to perform the repair work of the coding block and its signature on the failing server to ensure the data availability of users when they are not online. As the basis of the PARCC scheme, a new signature mechanism is designed based on the BLS signature, which enables the data owner to complete the generation of signature set while executing the regenerative code coding, thus improving the efficiency of signature generation.
References

[1] Kan Yang, Xiaohua Jia. An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2013, 24 (9): 1717-1726.

[2] Yujue Wang, Qianhong Wu, Bo Qin, et al. Identity-Based Data Outsourcing With Comprehensive Auditing in Clouds [J]. IEEE Transactions on Information Forensics and Security, 2017, 12 (4): 940-952.

[3] Mehdi Sookhak, F. Richard Yu, Albert Y. Zomaya. Auditing Big Data Storage in Cloud Computing Using Divide and Conquer Tables [J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29 (5): 999-1012.

[4] Yong Yu, Jianbing Ni, Man Ho Au, et al. Comments on a Public Auditing Mechanism for Shared Cloud Data Service [J]. IEEE Transactions on Services Computing, 2015, 8 (6): 998-999.

[5] Luo Yuchuan, Fu Shaojing, Xu Ming, et al. Enable data dynamics for algebraic signatures based remote data possession checking in the cloud storage [J]. China Communications, 2014, 11 (11): 114-124.

[6] Boyang Wang, Baochun Li, Hui Li. Oruta: privacy-preserving public auditing for shared data in the cloud [J]. IEEE Transactions on Cloud Computing, 2014, 1 (2): 43 – 56.

[7] Chang Liu, Rajiv Ranjan, Chi Yang, et al. MuR-DPA: Top-Down Levelled Multi-Replica Merkle Hash Tree Based Secure Public Auditing for Dynamic Big Data Storage on Cloud [J]. IEEE Transactions on Computers, 2015, 64 (9): 2609-2622.

[8] Chang Liu, Jinjun Chen, Laurence T. Yang, et al. Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-Grained Updates [J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25 (9): 2234 – 2244.

[9] Taek-Young Youn, Ku-Young Chang, Kyung-Hyune Rhee, et al. Efficient Client-Side Deduplication of Encrypted Data With Public Auditing in Cloud Storage [J]. IEEE Access, 2018, 6: 26578 – 26587.

[10] Dimakis A G, Godfrey P, Wu Y, et al. Network coding for distributed storage systems [J]. Information Theory, IEEE Transactions on. 2010, 56 (9): 4539–4551.

[11] Chen B, Curtmola R, Ateniese G, et al. Remote data checking for network coding-based distributed storage systems [C]. In Proceedings of the 2010 ACM workshop on Cloud computing security workshop. 2010: 31–42.

[12] Chen H, Lee P. Enabling Data Integrity Protection in Regenerating-Coding-Based Cloud Storage: Theory and Implementation [J]. Parallel and Distributed Systems, IEEE Transactions on. 2014, 25 (2): 407–416.