SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2024, Vol. 67, Iss. 3, 132102:1–132102:15
https://doi.org/10.1007/s11432-022-3597-3

Practical cloud storage auditing using serverless computing

Fei CHEN[1], Jianquan CAI[1], Tao XIANG[2*] & Xiaofeng LIAO[2]

[1]*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen* 518000*, China;*
[2]*College of Computer Science, Chongqing University, Chongqing* 400044*, China*

Received 14 January 2022/Revised 20 June 2022/Accepted 9 October 2022/Published online 19 October 2023

Abstract Cloud storage auditing research is dedicated to solving the data integrity problem of outsourced storage on the cloud. In recent years, researchers have proposed various cloud storage auditing schemes using different techniques. While these studies are elegant in theory, they assume an ideal cloud storage model; that is, they assume that the cloud provides the storage and compute interfaces as required by the proposed schemes. However, this does not hold for mainstream cloud storage systems because these systems only provide `read` and `write` interfaces but not the `compute` interface. To bridge this gap, this work proposes a serverless computing-based cloud storage auditing system for existing mainstream cloud object storage. The proposed system leverages existing cloud storage auditing schemes as a basic building block and makes two adaptations. One is that we use the read interface of cloud object storage to support block data requests in a traditional cloud storage auditing scheme. Another is that we employ the serverless computing paradigm to support block data computation as traditionally required. Leveraging the characteristics of serverless computing, the proposed system realizes economical, pay-as-you-go cloud storage auditing. The proposed system also supports mainstream cloud storage upper layer applications (e.g., file preview) by not modifying the data formats when embedding authentication tags for later auditing. We prototyped and open-sourced the proposed system to a mainstream cloud service, i.e., Tencent Cloud. Experimental results show that the proposed system is efficient and promising for practical use. For 40 GB of data, auditing takes approximately 98 s using serverless computation. The economic cost is 120.48 CNY per year, of which serverless computing only accounts for 46%. In contrast, no existing studies reported cloud storage auditing results for real-world cloud services.

Keywords cloud storage auditing, serverless computing, object storage, usability

1 Introduction

In recent years, with the popularity of 5G and the rapid development of the Internet of Things, the total amount of social data has rapidly increased. Whether it is a business or an individual user, there is a large amount of data that needs to be computed and stored. Cloud storage services have the advantages of on-demand payment, easy sharing, and cross-platform access. In addition, cloud computing has the advantages of flexible deployment and elastic expansion. These characteristics are attracting an increasing number of individuals, enterprises, and government organizations to adopt cloud services.

While cloud storage and cloud services are developing rapidly, security concerns also emerge. It has been reported from time to time that cloud services encountered data breaches and service breakdowns [1]. As such, users are beginning to worry about the security of their outsourced data. To address cloud data security issues, researchers have put forward the concept of cloud storage auditing, which has been extensively studied. Cloud storage auditing solves the problem of how remote users can be sure of the integrity of the data stored in the cloud. The value of such research is that even if the cloud tries to cover up data corruption, users can still leverage a small data proof to detect cloud data corruption.

As shown in Figure 1, the classical cloud storage auditing model has two entities, namely, user and cloud [2–6]. Before outsourcing the data to the cloud, the user first preprocesses the data using some secret key. The purpose is to embed secret authentication information in the outsourced data. Subsequently,

* Corresponding author (email: txiang@cqu.edu.cn)

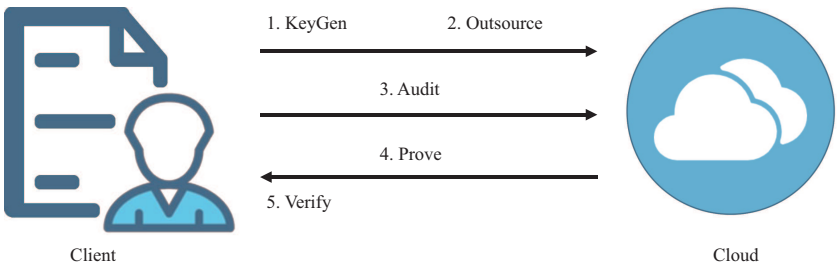

Figure 1 (Color online) Cloud storage auditing model.

the user stores the processed data in the cloud. To validate the integrity of the outsourced data, the user asks the cloud to return an integrity proof that authenticates the outsourced data. After receiving the data proof, the user obtains the up-to-date status of the outsourced data by checking whether the returned proof/authentication is correct.

Although considerable achievements have been made in current research on cloud storage auditing, most of the research aims at theoretical scheme construction. These theoretical schemes assume an ideal cloud service; that is, that the cloud provides `read`, `write`, and `compute` interfaces for storage. However, for the mainstream cloud object storage (COS) service, the compute interface, which is required by existing theoretical cloud storage auditing schemes, is not provided. Therefore, research on the practical usability of these schemes in current mainstream cloud practices has yet to be conducted. In practice, in addition to architecture and implementation issues, the economic cost is also a key factor that impacts the practical attraction of a cloud storage auditing scheme.

On the practical side, serverless computing has developed rapidly [7–9]. Compared with traditional virtual machine-based cloud computing, serverless computing does not require the user to manage a cloud computing infrastructure. It only uses a pay-as-you-go charging model, which considerably saves the user's cost in case the user's computation need is small. This feature is suitable for cloud storage auditing and contributes to its large-scale application due to economic attraction. From the research perspective, the serverless computing paradigm provides a good solution for the prototype verification of cloud storage auditing research. From the cloud service providers' perspective, the application of cloud storage auditing protocols will help them provide better cloud storage services and prove to consumers that their cloud services are reliable. From the cloud users' perspective, reducing the economic cost and operation cost when deploying an auditing system is helpful.

Our work. The purpose of this work is to promote cloud storage auditing from theory to practice. To this end, we designed and implemented a cloud storage auditing system by leveraging the serverless computing paradigm. The proposed system is built on existing cloud storage auditing protocols in a black box manner. Thus, this work supplements existing research by making them practical.

The main design goal of the proposed system is to be usable, efficient, and affordable. That is, the proposed system should support existing cloud storage upper-layer applications, e.g., file preview and file sharing. The system should also run quickly. An auditing query should be finished in a small amount of time. The system should also be cost-effective so that average users are affordable for such a system. Our architectural and technical design to achieve these goals is detailed later in the main text.

To support the design goals, the proposed system works roughly as follows. When the system initializes, a user employs a client program to outsource data to the cloud. The user chooses a traditional, efficient cloud storage auditing protocol. The user first embeds secret authentication information in the data, called the authentication tag. Then, the user outsources the processed data using public interfaces provided by COS, which is the mainstream cloud storage service on the market. The user also deploys a cloud service, called serverless cloud function (SCF), to conduct computation in the cloud as needed in a traditional cloud storage auditing protocol.

After initialization, the user can run cloud storage auditing regularly. To send an auditing challenge, the user sends a challenge to the SCF. The SCF requests the challenged data blocks as well as the corresponding authentication tags using the read interface of COS. Upon obtaining the data, the SCF combines these data as well as the authentication tags to obtain an aggregated one. The SCF later sends back the aggregated data block and its authentication tag to the user as a proof. Upon receiving the proof, the user checks whether the proof is correct and concludes whether the outsourced data are intact.

In addition to the proposed system, we also built a cost model to understand the economic cost of a

cloud storage auditing system and evaluate the practical feasibility of the proposed system.

We conducted a series of performance tests on the proposed system. For each different dataset, we conducted multiple tests to obtain the actual performance of the various algorithms involved. The experimental results show that the proposed system based on serverless computing is stable and efficient. Through the cost model, we obtained the system overhead for different datasets deployed on different mainstream cloud platforms. The results show that the proposed cloud storage auditing system is cost-effective and conducive to reducing the expenses of practical cloud storage auditing.

Contribution. To summarize, the contribution of the paper is as follows.

• We propose using serverless computing to make cloud storage auditing protocols practical. To implement this idea, we propose an architecture for practical cloud storage auditing. We also show an economic model to evaluate the cost of a practical storage auditing system.

• We prototype and open-source the proposed system. The proposed system supports all existing mainstream cloud services. The experimental results also validate the practical usability, efficiency, and affordability of our proposed model.

2 Related work

Cloud storage auditing. The cloud storage auditing problem was first raised and solved by Juels et al. [10] and Ateniese et al. [11] in 2007. Juels et al. [10] proposed the "proofs of retrievability" method, referred to as PoR. PoR first divides the data into blocks and then randomly inserts secret blocks between the original data blocks, which are called sentinels. Subsequently, PoR shuffles the data together with the sentinels and uploads it to the cloud. When auditing, the cloud is required to return specified data blocks. It can be proven probabilistically that the PoR scheme is secure; its security is based on the secret sentinels and their insertion positions.

Simultaneously, Ateniese proposed the "provable data possession" method, called PDP. Similar to PoR, PDP first divides the data into blocks and calculates the authentication tag for each data block. Unlike PoR, which uses random insertion and permutation, in PDP, the user uploads the original data blocks and the authentication tags to the cloud storage. When an auditing is initiated, the user requires the cloud to calculate an aggregated challenged data block and an authentication tag in a well-defined manner. After receiving the aggregated proof data, the user uses a calculation similar to the one used to calculate the authentication block to verify whether the outsourced storage is intact. The security of the PDP scheme is based on the security of cryptographic homomorphic signature or homomorphic message authentication.

Following the work of Juels and Ateniese, researchers expanded cloud storage auditing research according to different scenarios and needs. The first follow-up work was to support data dynamics. A significant drawback of PDP and PoR is that they only support static data. Regardless of whether it is a PDP or a PoR solution, most cloud storage auditing solutions divide the data into blocks. The difficulty in supporting data dynamics lies in the following fact: to resist replay attacks, authentication information must be bound to the index of the data block, such as calculating authentication tags in PDP and the position information of secret blocks in PoR. When the data change dynamically, the authentication information based on the original index will become invalid and will have to be recomputed.

To solve this problem, researchers have proposed various methods. A classic solution is to use the "skip list" [12]. It introduces the concept of "rank" instead of directly using the index of the data block to resist replay attacks. At the same time, "rank" also contains logical information about the mutual position of data blocks. When updating data, the relationship between data blocks can be maintained with less overhead. Another solution is to use an authentication data structure similar to the Merkle Hash tree (MHT). The MHT was first proposed by Merkle [13]; its structural characteristics are widely used in result verification. In this direction, Wang et al. [14] proposed a cloud storage scheme based on an MHT authentication structure. When the data are updated, the server recalculates the root value of the MHT tree, and the client verifies the old value. Similarly, the root value contains the logical information of the index between the data blocks. If the verification passes, the client accepts the new root and signs it. In addition to these two approaches, data dynamics can also be achieved by establishing a table that dynamically maintains the index changes between data blocks. Generally, this method is more efficient while requiring more storage overhead; it was also adopted in existing research, e.g., [15].

Later, with the expansion of the application scale of cloud services, researchers studied cloud storage

auditing that supports public auditing and privacy protection. By introducing a more capable third party, it is possible to deal with heavy auditing tasks more effectively and to ease users from auditing tasks [16, 17]. Usually, such an entity is called a trusted third party (TPA). Many TPA solutions supporting public auditing are based on cryptographic homomorphic techniques. In a homomorphic message authentication solution, a data owner asks the TPA to assist it in completing the integrity auditing task by granting the TPA critical metadata information, including message authentication tags and corresponding public keys [14, 18, 19].

Homomorphic aggregator approaches work similarly. The data owner asks the TPA to calculate the authentication information and lets the TPA complete the auditing task [20, 21]. The introduction of third parties has brought benefits and new problems; that is, the security of third parties makes user data privacy a new challenge. For example, in a scheme using a holomorphic technique, TPA can obtain the original data information through a linear combination of the original blocks during the sampling inspection process. To solve this problem, researchers have proposed many privacy-based public auditing schemes [22]. In addition, existing research also leveraged user identity to ease key management for storage integrity auditing [23–25]. Recent theoretical research also explores new techniques and new methods to support cloud storage auditing [26–33].

To summarize, existing research has extended cloud storage auditing to support versatile functionalities and have different security foundations. However, these studies are theoretical in nature as they assume the cloud storage service supports user-defined computation. This assumption does not hold in the real world. The existing mainstream cloud storage service is the COS service, which does not provide such an interface. This makes existing schemes ineffective in practice. In addition, existing schemes do not consider user cost, which is an influential factor when using cloud services in the real world. In this work, we aim to solve these practical gaps from a real-world perspective.

Serverless computing. In recent years, cloud serverless computing technology has gradually matured and been extensively studied [7, 30, 34]. Mainstream cloud service providers also began to provide serverless computing services.

Serverless computing is a hosted cloud computing service driven by events. It has the characteristics of fine-grained user management and realizes the characteristics of billing according to practical resource usage and elastic scaling [9, 35]. This is a good fit for the challenge-response auditing pattern in cloud storage integrity checking. When deploying traditional cloud storage auditing, it is necessary to rent a cloud virtual machine server. This approach is considered cumbersome because it may not fully realize the potential of the cloud server. Using serverless computing, the cloud service provider will allocate resources as needed for one run of an auditing task, which realizes on-demand execution and on-demand charging.

3 Preliminary

3.1 System specification

We employ traditional notation to specify a cloud storage auditing protocol. The protocol has five components and runs in two phases. The first is the data outsourcing phase. It runs as follows:

• $\mathsf{KeyGen}\left(1^{\lambda}\right) \rightarrow K$. The user first runs this algorithm to generate a secret key K according to the security level λ. The user keeps the key locally.

• $\mathsf{Outsource}\left(F, K\right) \rightarrow F'$. Later, the user employs the secret key k to process the data F. The processing embeds a secret authentication tag for each data block of F and then obtains the processed data F'. After the data processing is completed, the user uploads the processed data to the cloud storage.

Then, the user can regularly audit the integrity of the outsourced storage in the audit and verification phase. It runs as follows:

• $\mathsf{Audit}\left(K\right) \rightarrow q$. The user employs this algorithm to generate an auditing challenge to the cloud. The user then sends the challenge to the cloud, asking the cloud to prove that the outsourced data are intact.

• $\mathsf{Prove}\left(F', q\right) \rightarrow (\chi,\ \Gamma)$. After receiving an auditing challenge from the user, the cloud uses this algorithm to generate a proof (χ, Γ). Then, the cloud returns the proof to the user.

• $\mathsf{Verify}\left(\chi, \Gamma, q, K\right) \rightarrow \delta$. To finish the auditing challenge, the user employs this algorithm to verify whether the proof returned from the cloud is correct using its secret key. If the outsourced data are intact, the output is $\delta = 1$. Otherwise, the output is $\delta = 0$, indicating that the data are damaged in the

cloud.

A cloud storage auditing system can be implemented based on the theoretical model above. The traditional cloud storage auditing protocol has only two executive bodies: user and cloud. The theoretical model assumes that cloud service vendors provide both storage services and corresponding computing capabilities over the storage. However, it should be noted that current practice deviates from the assumption. Storage and compute services are two different, independent services; no direct computing capability is provided over the outsourced storage. Therefore, when deploying a cloud storage auditing system, we should also consider how to choose a suitable cloud compute service in addition to choosing a cloud storage service.

3.2 Design requirements and challenges

In this subsection, we summarize the design requirements and challenges of a practical, usable cloud storage auditing system.

First, the system should support and be able to run in all major cloud service providers. The current mainstream cloud service providers include AWS, Azure, Alibaba Cloud, and Tencent Cloud. To better attract new users, these mainstream cloud platforms provide standardized and similar operating specifications. Users can easily get started and complete their business when using related cloud services or migrating their systems to different cloud platforms. This feature is also the reason for the rapid expansion of the cloud service market in recent years and the increasingly fierce competition. Therefore, when designing a cloud storage auditing system, we should support mainstream cloud service providers and mature cloud services.

Second, the economic cost should be affordable. While enjoying convenient service from the cloud, users need to pay for storage and computation services. For users who are cautious about security, a storage auditing system incurs additional costs in addition to the basic storage costs. Depending on the additional cost, the users' behavior may be influenced to a large extent. To make cloud computing more attractive, a small auditing cost is more desired. To understand and minimize the cost, one needs to understand cloud computing at a reasonable depth.

Third, the system should be secure. Security is probably the most important factor when designing a cloud storage auditing system. Instead of designing a new auditing scheme, it is better to use a well-studied, existing scheme. Because its security is analyzed by many researchers, users can gain more confidence in security. However, a secure scheme may also be insecure when new attacks, such as quantum attacks, are found. In this case, the system should support easy replacement of a new, secure auditing scheme. We need to solve these challenges.

Fourth, the system should be usable. To achieve this goal, the cloud storage auditing system must solve two challenges. One is that the deployment of the system should not affect the use of other cloud services. For example, for cloud storage services, data preview and image processing are two widely used services. These services should continue to be usable after an auditing system is deployed. The other is system performance. System performance is a key factor for cloud storage auditing protocols to become practical.

3.3 Serverless computing basics

The SCF is a kind of cloud computing service that only charges a user for the actual resource consumption per run. It supports invoking user-deployed programs when user-defined conditions are met. In this case, the cloud automatically allocates resources to run the user's program, which does not require the user to maintain some resources all the time.

When using SCF, users only need to pay attention to the function interface specifications of the underlying cloud service providers when designing and implementing cloud functions. A cloud function is the basic unit of resource scheduling and program running. Its internal logic is determined by the user. This is no different from a user writing a program locally. Mainstream cloud platforms support cloud functions with many programming languages.

After the programming work of an SCF is finished, the SCF can be deployed to the cloud. To invoke the SCF, the user needs to configure related triggers. A trigger activates the execution of the cloud function. The trigger specifies the running conditions of the cloud function. At present, many cloud service providers support a variety of trigger options to meet users' needs in different business scenarios. Any behavior that can generate an event and invoke the execution of a cloud function could be set as a

trigger. In current practice, there are API (application programming interface) gateway triggers, timing triggers, and COS triggers. Users can choose an appropriate one for a cloud function.

In addition to starting the execution of the cloud function, the trigger also passes the event as an input parameter to the cloud function. An event is a data structure designed by a cloud service provider, e.g., JSON format. When triggering the function, the user can choose a synchronous or asynchronous mode. Finally, the event is passed to the cloud function as an input parameter.

4 Proposed system

4.1 Main idea

At a high level, we propose to combine existing theoretical cloud storage auditing schemes with mainstream cloud services in a workable, efficient, and economical way. In the following, we explain the details by showing how to address the design challenges in Subsection 3.2.

For the first challenge, we achieve the support of existing mainstream cloud services by using common COS and SCF services. More specifically, we employ COS to host the user's outsourced data and SCF to conduct computing services for theoretical cloud storage auditing schemes. COS is the most common cloud storage service today. It is supported by all main cloud vendors. The SCF is a computing service that has become popular in recent years. For the user, it eases the burden of managing cloud resources manually, for instance, managing a virtual machine. SCF is also supported by main cloud vendors.

For the second challenge, we reduce the cost of a storage auditing system from two aspects. First, instead of using the traditional cloud virtual machine to support the computing capability, we use SCF to support the 'prove' algorithm in the traditional storage auditing scheme. SCF is more lightweight than a virtual machine. It only charges the user for each run of the program but does not charge the user to rent a virtual machine for a fixed, long period. This enables a smaller cost. Second, we deploy COS and SCF in the same geographic region of the same cloud service provider to reduce costs, which will bring a few benefits. The communication cost in the same region is free because it belongs to the local area network of the cloud vendor and does not involve data transfer over the public internet. In addition, COS and SCF also have better communication efficiency. Cloud service providers generally optimize or provide dedicated data communication lines for their own products. This improves the execution efficiency and stability of the system.

For the third challenge, we design the system in a modular manner and reduce the security of the proposed system to the underlying theoretical auditing scheme. The proposed method takes a theoretical PDP-style cloud storage auditing as a black box that can be plugged into the system as a module. Thus, existing theoretical schemes can be used. The security of the proposed system also follows from the security of the adopted theoretical auditing scheme that is well studied. In this regard, users have the control to choose a proper auditing scheme and change a scheme when needed in the future.

For the fourth challenge, we maintain the outsourced data with the original format and separate the authentication information as an independent data object. Because the outsourced original data and authentication tags are separated, all existing cloud upper layer applications are preserved and supported. Moreover, we keep the cost of the proposed system small by using SCF and adopting the same-region architecture. This further contributes to the better usability of the proposed system.

4.2 System architecture and implementation overview

Figure 2 shows the architecture of the proposed cloud storage auditing system based on the serverless computing paradigm and COS. It consists of three main entities, i.e., user, SCF, and COS. Among them, SCF and COS are deployed under the same region of the same cloud. SCF provides the computing service that is required in a theoretical auditing scheme and can be calculated on demand and flexibly expanded. COS provides the storage service through its `read/write` interfaces using HTTP(S) requests/responses.

The system works as follows. First, the user generates a secret key using the KeyGen algorithm on the user's local machine. Then, the Outsource algorithm is used to process the data. The processed data have two parts, i.e., the original data and the corresponding authentication tags for each data block. The user then sends the two parts to the COS as two independent objects using the cloud's API. This finishes the system setup.

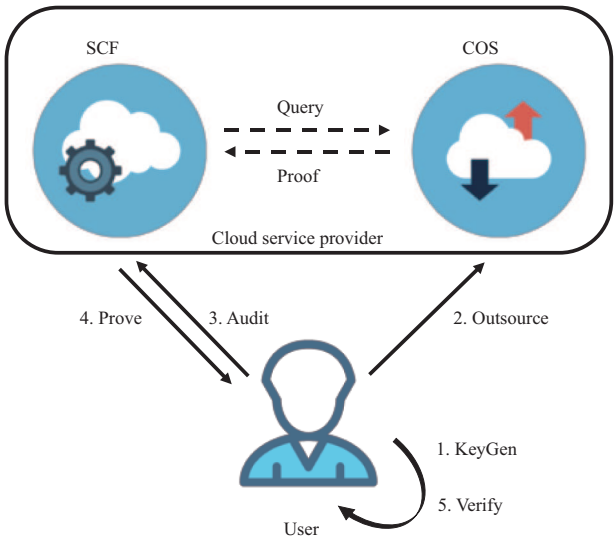

Figure 2 (Color online) System architecture.

Later, the user can conduct regular storage auditing. To issue an audit query, the user executes the Audit algorithm and calculates the challenge, which is further sent to SCF using the cloud API. In detail, the user sends the challenge data as the body content of an HTTPS request according to the input specification of SCF. The cloud serverless computing API gateway uniformly processes the received HTTPS requests. The gateway further converts the content of the HTTPS request into the input event of the SCF and triggers the cloud function.

Once triggered, SCF starts to execute. The challenge data are first parsed from the SCF input event. Subsequently, the SCF downloads the specified data and authentication tags of the challenged data blocks from the cloud storage through the COS API. After that, SCF runs the Prove algorithm to generate an auditing proof. Later, SCF returns the proof in a specified data format to the user who initially sent the auditing request to the SCF. The client receives the HTTPS RESPONSE returned by the SCF and then parses the auditing proof data. Finally, the user executes the Verify algorithm to check whether the cloud data are intact.

Based on the system architecture, we provide more technical details of the cloud storage auditing system we implemented. We chose Tencent Cloud as a representative backend cloud service provider. It is among the mainstream cloud service providers such as Amazon, Azure, and AliCloud. And its cloud services are similar to others. We note that any mainstream cloud can be used.

We used Java to implement a prototype of the proposed system. The implementation has two executable programs. One is on the user side. It implements the KeyGen, Outsource, Audit, and Verify algorithms. The other is on the cloud in the form of SCF. It implements the Prove algorithm. The SCF is executed after being triggered by the user. To invoke SCF, we used a cloud API gateway as the trigger. It first receives auditing challenge data from the user program in an API call. It then parses the challenge data and invokes SCF. SCF later continues to run the Prove algorithm. We do not use any program on the COS. It is used as an infrastructure to provide cloud storage services and supports data accesses for the two executable programs.

4.3 Adopted theoretical auditing scheme

We employ an improved theoretical cloud storage auditing scheme of [15, 36] as the building block in our implementation, because it is one of the state-of-the-art methods. In addition, this method supports partial data recovery, which promotes usability. Figure 3 shows the details. We note that any PDP-style theoretical scheme could be used.

To make this paper self-contained, we briefly explain the theoretical scheme. We first explain the symbols. The scheme works on a Galois Field GF(q), where $q = 2^8$. $F_{K_1}(\cdot)$ represents a pseudorandom function, which maps the input to an element over the Galois field. We used the advanced encryption standard (AES) algorithm as the random function in the implementation. The matrices C, H are used

- Setup (public parameters)
  GF(q), (n, k) Reed Solomon error correcting code
- $\mathsf{KeyGen}(1^\lambda) \to K$
  $s = (s_1, \ldots, s_{n-k}) \leftarrow \mathrm{GF}(q)^{n-k}$
  K_1 for PRF $F_{K_1}(\cdot) : \mathbb{Z} \to \mathrm{GF}(q)^{n-k}$
  $\alpha_1, \ldots, \alpha_n \xleftarrow{R} \mathrm{GF}(q)$, $[C, H]$ as in Eq. (1)
  $K = (K_1, s, \alpha_1, \ldots, \alpha_n)$
- $\mathsf{Outsource}(F, K) \to F'$
  $F \leftarrow (m_1, m_2, \ldots, m_{n_F})$ where $m_i \in \mathrm{GF}(q)^k$
  $\sigma_i \leftarrow (m_i C^{-1} H) \circ s + F_{K_1}(i)$
  // $\circ$ denotes vector entrywise product
  $F' = (m_i, \sigma_i)$ where $i = 1, \ldots, n_F$
- $\mathsf{Audit}(K) \to q$
  $i_j \xleftarrow{R} [1, n_F]$ for $j = 1, 2, \ldots, l$
  $c_j \xleftarrow{R} \mathrm{GF}(q)$ for $j = 1, 2, \ldots, l$
  $q = (i_j, c_j)$ where $j = 1, \ldots, l$
- $\mathsf{Prove}(F', q) \to (\chi, \Gamma)$:
  $\chi \leftarrow \sum_{j=1}^{l} c_j \cdot m_{i_j}$
  $\Gamma \leftarrow \sum_{j=1}^{l} c_j \cdot \sigma_{i_j}$
- $\mathsf{Verify}(\chi, \Gamma, q, K) \to \delta$
  $\delta \leftarrow \Gamma \stackrel{?}{=} (\chi \cdot C^{-1} H) \circ s + \sum_{j=1}^{l} c_j \cdot F_{K_1}(i_j)$

Figure 3 Adopted theoretical auditing scheme.

to embed Reed Solomon error correcting codes and are defined as follows:

$$[C, H] = \left[\begin{array}{ccc|ccc} \alpha_1^0 & \cdots & \alpha_k^0 & \alpha_{k+1}^0 & \cdots & \alpha_n^0 \\ \alpha_1^1 & \cdots & \alpha_k^1 & \alpha_{k+1}^1 & \cdots & \alpha_n^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{k-1} & \cdots & \alpha_k^{k-1} & \alpha_{k+1}^{k-1} & \cdots & \alpha_n^{k-1} \end{array}\right]. \tag{1}$$

To outsource the data, the user chooses a Galois field $\mathrm{GF}(q = 2^8)$ and a (n, k) Reed Solomon error correcting codes, where n denotes the total code length and k denotes the message length. Then, each message has a parity code with length $n-k$. To generate the secret key, the user chooses $s = (s_1, \ldots, s_{n-k})$ in the Galois field and a secret key K_1 for the pseudorandom function. In addition, the user chooses the secret encoding matrix C, H as in (1). To outsource the data F, the user divides it into blocks $F = (m_1, m_2, \ldots, m_{nF})$, where m_i represents the ith data block and is a vector with length k. For each block, the user computes its authentication tag $\sigma_i \leftarrow (m_i C^{-1} H) \circ s + F_{K_1}(i)$. In essence, the tag is a combination of an error correction code and a message authentication code. For interested readers, please refer to [15] for more details.

To audit the integrity of the outsourced data on the cloud, the user chooses l random indices and corresponding coefficients $q = (i_j, c_j)$ as an auditing query. The user sends the query to the cloud. After receiving the challenge, the cloud computes two parts. One part is the linear combination of the challenged data, i.e., $\chi = \sum_{j=1}^{l} c_j \cdot m_{ij}$. Another part is the corresponding authentication information $\Gamma = \sum_{j=1}^{l} c_j \cdot \sigma_{ij}$. Then, the cloud sends the data proof (χ, Γ) back to the user. After receiving the proof, the user checks whether the following equation holds:

$$\Gamma \stackrel{?}{=} (\chi \cdot C^{-1} H) \circ s + \sum_{j=1}^{l} c_j \cdot F_{K_1}(i_j). \tag{2}$$

If the equation holds, the outsourced data are stored in the cloud intact; the auditing result is $\delta = 1$. Otherwise, the data are broken and output $\delta = 0$.

4.4 Cost analysis

In practice, users expect to understand the cost of a cloud storage auditing system. An affordable system is more desired. To better understand the cost, we analyze the cost components and determine the cost expression according to real world charging standards.

In general, the cost of the proposed system has two parts:

$$M = M_f + M_s. \tag{3}$$

Table 1 Symbols and definitions

Symbol	Description	Unit
p_{ru}	SCF resource usage fees	CNY/GB · s/month
p_{iv}	SCF invocation fees	CNY/10000 requests
p_{pn}	SCF public network downstream traffic	CNY/GB
j	Number of audits per day	–
r_m	SCF running memory	GB
r_t	SCF running time	s
l	Challenge data length	–
ds	The size of outsourced data	GB
p_{uf}	Standard storage usage fee	CNY/GB/month
p_{rf}	Standard read/write request fee	CNY/10000 requests
D	The size of the source data	GB
n	Block number of the source data	–
λ	The size of the authentication tags	GB

M represents the total monetary cost of the proposed cloud storage auditing system. M_f is the total expenditure of cloud functions per year. M_s is the total cost of object storage per year. Table 1 gives some symbols and their definitions used in the cost model. In the following, we show how to compute M_f and M_s in detail.

We first analyze SCF cost. Mainstream cloud service providers charge cloud functions from four respects.

• M_{ru} (resource usage). The cloud charges for the exact resource usage of each SCF running. The resource usage is obtained by multiplying the function memory size and the run time of the function. The measurement unit is gigabyte times second.

• M_{iv} (invocations). Each trigger execution of the function is recorded as one call. The cloud charges the total number of callings. The measurement is in 10000 invocations.

• M_{pn} (public network outbound traffic). The cloud also charges for outgoing network traffic. The measurement is in gigabytes. We note that the network traffic inside the cloud data center is free. That is, the communication between the SCF and COS is free.

• M_{ipc} (idle concurrency resource). The cloud function supports concurrent execution by specifying the number of total running instances. However, when the actual running instances are smaller due to smaller SCF invocations, the idle concurrency resource is also charged. The charging standard is similar to resource usage charging; the unit is also gigabyte times seconds.

Using the above charging standard, we obtain the detailed cost as follows:

$$M_f = M_{\mathrm{ru}} + M_{\mathrm{iv}} + M_{\mathrm{pn}} + M_{\mathrm{ipc}}, \tag{4}$$

$$M_{\mathrm{ru}} = (r_m/1024) \times (r_t/1000) \times j \times 365 \times p_{\mathrm{ru}}, \tag{5}$$

$$M_{\mathrm{iv}} = j \times \frac{365}{10000} \times p_{\mathrm{iv}}, \tag{6}$$

$$M_{\mathrm{pn}} = (D/n + \ \lambda) \times p_{\mathrm{pn}} \times j \times 365. \tag{7}$$

The proposed system does not cause M_{ipc} cost. We have $M_{\mathrm{ipc}} = 0$. Thus, we obtain the final expression of M_f as follows:

$$M_f = \left[\frac{r_m \cdot r_t \cdot p_{\mathrm{ru}}}{1024 \times 1000} + \frac{p_{\mathrm{iv}}}{10000} + \frac{(D + \lambda \cdot n) p_{\mathrm{pn}}}{n} \right] \times 365j. \tag{8}$$

We now analyze COS cost. In the usage scenarios of storage outsourcing, mainstream cloud platforms charge for COS services mainly from three aspects, i.e., storage usage fees, request fees, and network traffic fees. We denote them, respectively, as $M_{\mathrm{uf}}, M_{\mathrm{rf}}, M_{\mathrm{tf}}$. Thus, we have

$$M_s = M_{\mathrm{uf}} + M_{\mathrm{rf}} + M_{\mathrm{tf}}. \tag{9}$$

More specifically, the storage and request fees M_{uf} and M_{rf} can be computed as follows:

$$M_{\mathrm{uf}} = \mathrm{ds} \times p_{\mathrm{uf}} \times 12, \tag{10}$$

$$M_{\mathrm{rf}} = 2l/10000 \times j \times p_{\mathrm{rf}} \times 365. \tag{11}$$

For communication traffic, two scenarios exist. First, when the data are uploaded to the COS, the uploading communication is free for main cloud vendors. Second, auditing also incurs communication between SCFs and COSs. However, they belong to intranet traffic inside the data center because the proposed system deploys the SCF and COS in the same region. Its cost is also free. Thus, we have $M_{\mathrm{tf}} = 0$. Based on the above analysis, we can obtain the cloud storage cost as follows:

$$M_s = 12d_s \cdot p_{\mathrm{uf}} + \frac{2l \cdot p_{\mathrm{rf}}}{10000} 365j. \tag{12}$$

Finally, we obtain the total cost as

$$M = \left[\frac{r_m \cdot r_t \cdot p_{\mathrm{ru}}}{1024 \times 1000} + \frac{p_{\mathrm{iv}}}{1000} + (D/n + \lambda)p_{\mathrm{pn}} + \frac{2l \cdot p_{\mathrm{rf}}}{10000}\right] 365j + 12\mathrm{ds} \cdot p_{\mathrm{uf}}. \tag{13}$$

Using (13), we clearly see what factors affect the cost of the cloud storage auditing system and how they determine the final total cost. Intuitively, two factors that affect the cost are the cloud service provider's charging strategy and the user's usage strategy. For users, the cloud service provider's charging strategy is determined by the cloud, which is reflected in the fixed charging parameters of (13), e.g., $r_m, r_t, \ldots$. The users have no control of them. However, the users can optimize the total cost by tuning the total number of blocks and challenge length, i.e., n, l. Thus, Eq. (13) not only computes the total cost of a cloud storage auditing system but also gives an optimization perspective for users to reduce cost.

5 Performance evaluation

5.1 Setup

We implemented and open-sourced the proposed system using Java[1)]. The implementation used Tencent Cloud Object Storage[2)] to host the outsourced data and Tencent Serverless Cloud Function[3)] to provide the computing capability. For SCF, we configured 1 GB running memory and a single execution time limit of 900 s. In the user's local client, we use a computer configured with an AMD 4800U (1.8 GHz) CPU and 16 GB memory to run the client program. Because mainstream cloud service providers (e.g., Alicloud, Amazon) all support serverless computing, the proposed system could adapt to these cloud services directly.

For the employed theoretical auditing scheme [15,36], we chose GF(2^8) as the finite field. The advantage of this choice is that an element over the finite field is exactly the size of 1 byte, which is a basic data type in the Java language. Such a system is very efficient in computation, as it does not involve big integer arithmetic. In addition, we used ($n = 255$, $k = 223$) when setting the Reed Solomon error correction code. We divided the source data into blocks, where each block is 223 bytes and has a tag of size 32 bytes. We set the challenge length to $l = 460$ to obtain an auditing accuracy 99% [15].

5.2 Methodology

To evaluate the practical performance of the proposed system, we conducted multiple evaluations for different datasets. We conducted 10 tests for each dataset to obtain more stable performance results. In general, we measured the extra storage cost, communication cost, and computation cost of the proposed cloud storage auditing system.

For computational cost, we measured the performance of each component of the cloud storage auditing protocol. To observe the system performance in more detail, we divided the Outsource algorithm into two parts, i.e., data preprocessing and outsourcing. In data preprocessing, the user preprocesses the source data and generates the authentication tag for each data block. In outsourcing, the data and the tags are uploaded to the cloud. This part is greatly affected by the network.

In addition, to explore the economic cost of the proposed system, we also evaluated the monetary cost using the cost model in Section 4. Table 2 shows the charging strategy for cloud services of different mainstream cloud platforms. We used Tencent Cloud charging in the evaluation. We note that for SCF

1) Cai J. Sourcecode. 2021. https://github.com/szu-security-group/IntegrityCheckingUsingSCF.
2) https://cloud.tencent.com/product/cos.
3) https://cloud.tencent.com/product/scf.

Table 2 Mainstream cloud service fees (CNY)

Cloud service provider	Tencent	Ali-cloud	Amazon
COS usage (/GB/mon)	0.118	0.12	0.195
COS request (/10k requests)	0.01	0.01	0.015
SCF resource (/GBs)	0.00011108	0.000110592	0.00015047
SCF invocations fees (/10k)	0.0133	0.0133	0.018382
SCF outbound traffic (/GB)	0.8	0.5	0.7878

Table 3 Storage and communication cost

Data size	Storage cost (bytes)	Communication cost (bytes)
10 MB	1504704	255
20 MB	3009376	255
50 MB	7523424	255
100 MB	15046848	255
200 MB	30093664	255
500 MB	75234176	255
1 GB	154079552	255
1.5 GB	231119328	255

and COS charging, cloud service providers often offer certain free quotas for consumers per month to attract users. For example, for the SCF of Tencent Clouds, the free quota has reached 400000 GBs of resource usage and 1 million invocation times. This free quota is enough to meet the needs of most individual consumers. However, for generality and long-term usability, we do not consider the free quota.

As a short summary, we obtained the storage cost, preprocessing cost, auditing cost, verification cost, SCF computing cost, and SCF memory cost by direct measurement. We obtained the communication cost and monetary cost by numerical analysis. In Subsection 5.3, we show the detailed experimental results.

5.3 Result

Table 3 shows the storage and communication costs. The additional storage cost mainly lies in the authentication tags. In theory, the cost increases linearly with the dataset. The experimental results show that the ratio of extra storage cost to the outsourced data is approximately 14.3%. This is roughly consistent with the theoretical value.

The communication cost consists of bandwidth cost and HTTP request cost. Because the size of the auditing proof is determined by the block size and tag size, the bandwidth cost is fixed at 255 bytes. The HTTP request cost depends on the user's auditing strategy. When the number of user audits increases, the HTTP request cost also increases linearly.

The computational cost is shown in Table 4, which is also illustrated in Figure 4. From the table, we find that the computational costs of the KeyGen, Audit, and Verify algorithms are stable and small. The computing time of the data preprocessing increases linearly with the size of the dataset. It takes 74 s to compute the authentication tags of 1 GB of data. To generate auditing challenges and verify the cloud's response, the computation costs are roughly fixed at 0.2 and 95 ms. Both are very small. The time used for uploading the outsourced data to the cloud is greatly affected by network fluctuations. The internet speed we used was only 10 Mbps; it took a lot of time to upload data to the cloud for the first time. However, there is still no need to worry about the cost of this part. On the one hand, enterprise users have faster network lines. On the other hand, users that adopt cloud storage always need to upload the data. This is also a one-time cost.

For the SCF, the computation cost increases in a logarithmic manner. Our test results ranged from 17.5 to 69.2 s. The increased time comes from the increase in the average data requesting time of COS when the dataset becomes larger. We also found that the evaluation results of the same dataset fluctuate significantly. One reason comes from the randomness of the challenged data. Another reason is that the execution time of SCF is also affected by the cloud provider and the network. However, in general, the time spent on SCF is small. The memory cost is also acceptable.

Table 5 shows the monetary cost. We assumed that the auditing is conducted once a day and that the length of each auditing is 500 s, which is used for easy calculation with a successful auditing rate

Table 4 Computation cost

Size	KeyGen (ms)	Outsource (ms)	Audit (ms)	Verify (ms)	SCF time (ms)	SCF memory (MB)
10 MB	0.064	876	0.16	95.9	17455	158.7062
20 MB	0.058	1576	0.19	95.1	17171	124.6325
50 MB	0.066	3696	0.18	94.2	18786	146.1551
100 MB	0.066	7196	0.20	94.9	24076	107.2633
200 MB	0.063	14478	0.16	95.9	33852	97.5759
500 MB	0.061	35887	0.20	94.8	46899	80.9355
1 GB	0.064	74116	0.20	95.3	61498	78.3903
1.5 GB	0.060	121648	0.20	97.1	69245	84.6532

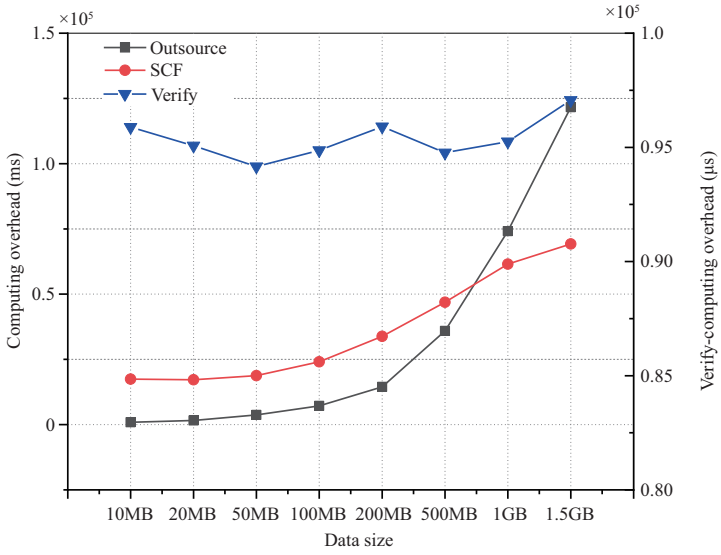

Figure 4 (Color online) Computation growth trend.

Table 5 Monetary cost (CNY/per year)

Data size (GB)	COS storage usage fees	COS request fees	COS total fees	SCF resource usage fees	SCF invocations fees	SCF public network outbound traffic fees	SCF total fees	Total fees
1	1.6192	0.1825	1.802	30.74024	0.04788	7.00378418	37.7919	39.594
5	8.096	0.1825	8.278	37.32179	0.04788	7.00378418	44.37345	52.652
10	16.192	0.1825	16.37	41.04018	0.04788	7.00378418	48.09184	64.466
15	24.288	0.1825	24.47	43.21529	0.04788	7.00378418	50.26696	74.737
20	32.384	0.1825	32.57	44.75856	0.04788	7.00378418	51.81023	84.377
40	64.768	0.1825	64.95	48.47695	0.04788	7.00378418	55.52861	120.48

larger than 99%. From the table, we find that the expenditure of the cloud storage auditing system is quantifiable, predictable and acceptable. As the dataset increases, the cost of COS always accounts for the most. The proportion of SCF expenses over the total cost gradually shrinks.

With Tencent Cloud as the cloud service provider, we implemented our auditing strategy for a maximum of 40 GB of data. The total annual expense is only 120.332 CNY, 46% of which is attributed to SCF accounts. In addition, the cost ratio of the SCF in the system decreases significantly with increasing outsourced storage, as shown in Figure 5. The specific expenditures of each item are listed in Table 5.

In addition, we also evaluated the cost model for different cloud service providers. The total cost results are shown in Figure 5. We find that the charging results on several mainstream cloud service providers are slightly different, but the trend is the same. From these results, it is reasonable to state that the proposed cloud storage auditing system is economical and practical.

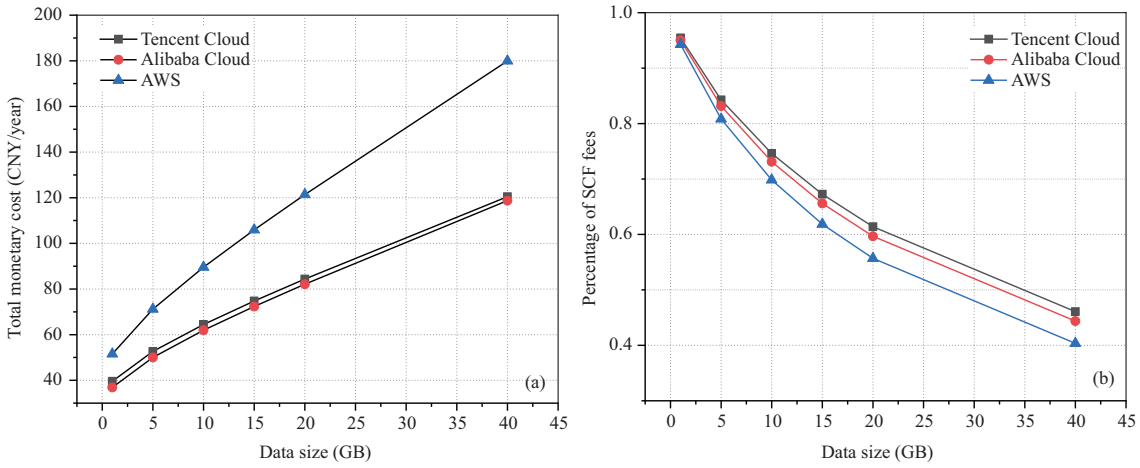

Figure 5 (Color online) Charging results of mainstream cloud service providers. (a) Total cost on different clouds; (b) SCF cost over total cost on different clouds.

6 Discussion

Security. Security is the most important requirement of the cloud storage auditing system. In the proposed system, two cloud services are used, i.e., COS and SCF. In the worst case, assuming that the cloud is malicious and includes malicious attacks from third parties, the proposed system is still secure. The reason is that it is the user that checks the auditing proof using the secret key. The security level of the system is still determined by the underlying theoretical cloud storage auditing scheme because the existing cloud storage auditing research generally assumes that the cloud is malicious. Thus, the security of the protocol is determined by the key stored locally. In the proposed system, COS can obtain cloud data, and SCF can compute the auditing proof. The data that can be obtained and computed by these capabilities are public. In other words, when a malicious cloud modifies, adds, or deletes the outsourced data or attempts to forge the data when computing the auditing proof, the proposed system can detect the malicious behavior.

Usability. As discussed in Section 4, the proposed system promotes usability from both the architecture perspective and the cost perspective. In addition to these two factors, we note that our open-source prototype also provides a starting point/template to implement a practical storage auditing system. Depending on the user's need, the user may choose another PDF-style auditing scheme to replace the implemented scheme. The user may also choose another cloud depending on its application requirement. Moreover, the experimental evaluation results show that the real economic cost is also small and affordable for existing mainstream cloud vendors. These two new perspectives also promote better usability.

Limitation, data dynamics and recovery. The proposed system treats the PDP auditing protocol as a black box. This means that as long as the PDP protocol adopted by the system has the ability to support data dynamics, the system can also support data dynamics. However, although the system can achieve data dynamics, the performance is not satisfactory. This is because COS does not support data insertion and deletion capabilities at the byte scale natively by providing APIs. For example, users can easily download COS data starting from a specified byte position. However, when the user wants to insert a new data block, the only accessible way is that the user downloads the outsourced data for modification and then uploads the new whole data again.

In addition, the theoretical protocol we used has data recovery capabilities. In addition to malicious cloud data damage due to attackers, cloud storage can also be damaged due to hardware failures. In this case, the proposed system can recover partial data after detecting a data damage occurrence. However, the data recovery capability is limited. When the damaged data are large, we cannot recover all the data. This is because error correcting is only used on the data block level, not on the data as a whole.

7 Conclusion

We have proposed a practically usable cloud storage auditing system. This is the first time that a cloud storage auditing system has been constructed using the serverless computing paradigm. We have

prototyped and evaluated the practical performance of the system using a mainstream cloud service provider. The evaluation shows that the proposed system is practically usable. We believe that using serverless computing is an ideal way to bring cloud storage auditing protocols from theory to reality. The proposed cloud storage auditing system based on the serverless computing paradigm and its cost model may also serve as a direction and reference for future cloud storage security research.

Acknowledgements This work was partially supported by National Natural Science Foundation of China (Grant Nos. 62272318, 61872243, 62072062, U20A20176), Shenzhen Science and Technology Program (Grant No. JCYJ20220531102802005), Guangdong Basic and Applied Basic Research Foundation (Grant No. 2020A1515011489), and Natural Science Foundation of Chongqing (Grant No. cstc2022ycjhbgzxm0031).

References

1 Millward W T. The 10 biggest cloud outages of 2021. CRN Cloud, https://www.crn.com/slide-shows/cloud/the-10-biggest-cloud-outages-of-2021-so-far-
2 Guo X J, Li J, Liu Z L, et al. Labrador: towards fair and auditable data sharing in cloud computing with long-term privacy. Sci China Inf Sci, 2022, 65: 152106
3 Chang J Y, Shao B L, Ji Y Y, et al. Secure network coding from secure proof of retrievability. Sci China Inf Sci, 2021, 64: 229301
4 Zhang Y H, Zhang T T, Xu S M, et al. Revocable and certificateless public auditing for cloud storage. Sci China Inf Sci, 2020, 63: 209302
5 Xue J T, Xu C X, Zhao J N, et al. Identity-based public auditing for cloud storage systems against malicious auditors via blockchain. Sci China Inf Sci, 2019, 62: 032104
6 Zhang R, Ma H, Lu Y, et al. Provably secure cloud storage for mobile networks with less computation and smaller overhead. Sci China Inf Sci, 2017, 60: 122104
7 Jonas E, Schleier-Smith J, Sreekanti V, et al. Cloud programming simplified: a berkeley view on serverless computing. 2019. ArXiv:1902.03383
8 Wen J, Chen Z, Liu Y, et al. An empirical study on challenges of application development in serverless computing. In: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2021. 416–428
9 Schleier-Smith J, Sreekanti V, Khandelwal A, et al. What serverless computing is and should become. Commun ACM, 2021, 64: 76–84
10 Juels A, Kaliski Jr B S. PORs: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, 2007. 584–597
11 Ateniese G, Burns R, Curtmola R, et al. Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, 2007. 598–609
12 Erway C C, Küpçü A, Papamanthou C, et al. Dynamic provable data possession. ACM Trans Inf Syst Secur, 2015, 17: 1–29
13 Merkle R C. Protocols for public key cryptosystems. In: Proceedings of IEEE Symposium on Security and Privacy, 1980. 122–122
14 Wang Q, Wang C, Li J, et al. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proceedings of European Symposium on Research in Computer Security, 2009. 355–370
15 Chen F, Meng F, Xiang T, et al. Towards usable cloud storage auditing. IEEE Trans Parall Distrib Syst, 2020, 31: 2605–2617
16 Shah M A, Swaminathan R, Baker M, et al. Privacy-preserving audit and extraction of digital contents. 2008. https://eprint.iacr.org/2008/186.pdf
17 Tian H, Chen Y, Jiang H, et al. Public Auditing for trusted cloud storage services. IEEE Secur Priv, 2019, 17: 10–22
18 Wang C, Chow S S M, Wang Q, et al. Privacy-preserving public auditing for secure cloud storage. IEEE Trans Comput, 2011, 62: 362–375
19 Li J, Zhang L, Liu J K, et al. Privacy-preserving public auditing protocol for low-performance end devices in cloud. IEEE Trans Inform Forensic Secur, 2016, 11: 2572–2583
20 Tian H, Chen Y, Chang C C, et al. Dynamic-Hash-table based public auditing for secure cloud storage. IEEE Trans Serv Comput, 2015, 10: 701–714
21 Zhang Y, Xu C, Liang X, et al. Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation. IEEE Trans Inform Forensic Secur, 2016, 12: 676–688
22 Wang C, Ren K, Lou W, et al. Toward publicly auditable secure cloud data storage services. IEEE Netw, 2010, 24: 19–24
23 Wang H, Wu Q, Qin B, et al. Identity-based remote data possession checking in public clouds. IET Inf Secur, 2014, 8: 114–121
24 Li J, Yan H, Zhang Y. Identity-based privacy preserving remote data integrity checking for cloud storage. IEEE Syst J, 2020, 15: 577–585
25 Lu X, Pan Z, Xian H. An integrity verification scheme of cloud storage for internet-of-things mobile terminal devices. Comput Security, 2020, 92: 101686
26 Qiu H, Noura H, Qiu M, et al. A user-centric data protection method for cloud storage based on invertible DWT. IEEE Trans

Cloud Comput, 2021, 9: 1293–1304
27 Nachiappan R, Javadi B, Calheiros R N, et al. Cloud storage reliability for big data applications: a state of the art survey. J Netw Comput Appl, 2017, 97: 35–47
28 Khan M T, Tran C, Singh S, et al. Helping users automatically find and manage sensitive, expendable files in cloud storage. In: Proceedings of the 30th USENIX Security Symposium, 2021
29 Li J, Wu J, Jiang G, et al. Blockchain-based public auditing for big data in cloud storage. Inf Process Manage, 2020, 57: 102382
30 Castro P, Ishakian V, Muthusamy V, et al. The rise of serverless computing. Commun ACM, 2019, 62: 44–54
31 Shen W, Qin J, Yu J, et al. Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Trans Inform Forensic Secur, 2018, 14: 331–346
32 Tan C B, Hijazi M H A, Lim Y, et al. A survey on proof of retrievability for cloud data integrity and availability: cloud storage state-of-the-art, issues, solutions and future trends. J Netw Comput Appl, 2018, 110: 75–86
33 Xiong S, Ni Q, Wang L, et al. SEM-ACSIT: secure and efficient multiauthority access control for IoT cloud storage. IEEE Int Things J, 2020, 7: 2914–2927
34 Sreekanti V, Wu C, Lin X C, et al. Cloudburst. Proc VLDB Endow, 2020, 13: 2438–2452
35 Baldini I, Castro P, Chang K, et al. Serverless computing: current trends and open problems. In: Proceedings of Research Advances in Cloud Computing, 2017
36 Wang H, Feng L, Ji Y, et al. Toward usable cloud storage auditing, revisited. IEEE Syst J, 2022, 16: 693–700