Future Generation Computer Systems 49 (2015) 58–67

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

External integrity verification for outsourced big data in cloud and IoT: A big picture

Chang Liu *, Chi Yang, Xuyun Zhang, Jinjun Chen

Faculty of Engineering and IT, University of Technology Sydney, Australia

HIGHLIGHTS

- Security of Big Data in cloud and IoT is becoming a major problem.
- Efficient external integrity verification is an important part of data security.
- We provide a big picture through summarizing and analysis of the main results of external integrity verification schemes for big data in cloud.

ARTICLE INFO

Article history:
Received 29 March 2014
Received in revised form
4 July 2014
Accepted 15 August 2014
Available online 27 August 2014

Keywords:
Cloud computing
Data security
Big data
Internet of Things
Integrity verification
Public auditing

ABSTRACT

As cloud computing is being widely adopted for big data processing, data security is becoming one of the major concerns of data owners. Data integrity is an important factor in almost any data and computation related context. It is not only one of the qualities of service, but also an important part of data security and privacy. With the proliferation of cloud computing and the increasing needs in analytics for big data such as data generated by the Internet of Things, verification of data integrity becomes increasingly important, especially on outsourced data. Therefore, research topics on external data integrity verification have attracted tremendous research interest in recent years. Among all the metrics, efficiency and security are two of the most concerned measurements. In this paper, we will bring forth a big picture through providing an analysis on authenticator-based data integrity verification techniques on cloud and Internet of Things data. We will analyze multiple aspects of the research problem. First, we illustrate the research problem by summarizing research motivations and methodologies. Second, we summarize and compare current achievements of several of the representative approaches. Finally, we introduce our view for possible future developments.

1. Introduction

Big data is attracting more and more interests from increasing groups of professionals from almost every industry. A few examples are oil and gas mining, scientific research (biology, chemistry, physics), online social networks (Twitter, Facebook), multimedia data, and business transactions. With mountains of data collected from increasingly efficient data collecting devices as well as stored on fast-growing storage hardware, people are keen to find solutions to store and process the data more efficiently, and to discover more values from the mass at the same time. When referring to big data research problems, people often brings the 4 V's—volume, velocity, variety, and value. These pose various brand-new challenges to computer scientists nowadays.

The recently emerged cloud computing, known to be the latest development across data center, parallel distributed systems and service computing technologies, is widely considered as the most promising technological backbone for solving big data problems [1]. The pay-as-you-go payment model of cloud can cut into the investments by enabling zero expense in setting up and maintaining of expensive computational and storage hardware, as well as provide on-the-fly problem solving. The services cloud can provide, ranging from SaaS (Software-as-a-Service), PaaS (Platform-as-a-Service), and IaaS (Infrastructure-as-a-Service), can offer solutions for big data problems from any level [2,3]. Cloud also offers elasticity and scalability which can result in further saving of costs in many practical applications involving fast-updating dynamic data. To date, large amounts of business data of numerous big companies have been moved into and managed by clouds such as Amazon AWS, IBM SmartCloud and Microsoft Azure. MapReduce

* Corresponding author. Tel.: +61 4 22701995.
E-mail addresses: changliu.it@gmail.com (C. Liu), chiyangit@gmail.com (C. Yang), xyzhanggz@gmail.com (X. Zhang), jinjun.chen@gmail.com (J. Chen).

http://dx.doi.org/10.1016/j.future.2014.08.007
0167-739X/

distributed computing framework is a core technique for cloud data processing [4–6].

Despite those stand-out advantages of cloud, there are still strong concerns regarding service qualities, especially data security and user privacy. There was much research in this area in recent years [2,4,7–14]. In fact, data security has been frequently raised as one of the top concerns in using cloud. In this new model, user datasets are entirely outsourced to the cloud service provider (CSP), which means they are no longer stored and managed locally. As CSPs cannot be deemed completely trusted, this fact brings several new issues. To name a few, first, when applied in cloud environments, many traditional security approaches will stop being either effective or efficient especially when handling big data tasks. Second, not only CSPs need to deploy their own security mechanisms (mostly conventional), but the clients also need their own verification mechanisms, no matter how secure the server-side security mechanisms claimed to be; the verifications may not bring additional security risks and must be efficient in computation, communication and storage in order to work in correlation with cloud and big data. Third, as the storage server is only semi-trusted, the client may be deceived by deliberately manipulated responses. All these new requirements have made the problem very challenging and therefore started to attract many computer science researchers' interest in recent years.

Internet of Things (IoT) is the next-generation computing platform that is integrated into our daily life [15]. The main idea is to connect every object with Internet technologies (embedded sensors, localization, communication, etc.) to transfer and process the data they produced in a real-timed fashion [16]. Currently, IoT related research is still in its early stage, as there is no unified standard for its design and implementation. Due to its real-timed nature, IoT datasets can be very large when data is generated over a certain amount of time. IoT data can also be put to cloud [17] for processing. Also, much of the IoT data needs to be protected carefully as they are very personal, such as location, health data, etc. Therefore, security in IoT data is also a very important research topic.

Data security includes many dimensions; the three main dimensions are confidentiality, integrity and availability. In this paper, we will focus on data integrity. Data integrity, means that data is needed to be maintained intact. Integrity verification and protection is an active research area; numerous research problems belong to this area have been studied intensively in the past. As a result, the integrity of data storage can now be effectively verified in traditional systems through the deployments of Reed–Solomon code, checksums, trapdoor hash functions, message authentication code (MAC), digital signatures, and so on. However, as stated above, the data owner (cloud user) still needs a method to verify their data stored remotely on a semi-trusted cloud server, no matter how secure the cloud claims to be. In other words, a cloud service provider must enable verifications from an external party that is independent to the cloud. The party could be the client herself, or a third party auditor. A straightforward approach is to retrieve and download from the server all the data the client wanted to verify. Unfortunately, when data size is large, it is very inefficient in the sense of both time consumption and communication overheads. Moreover, when the client needs a third party to verify the data on her behalf, all data will be exposed to the third party. To address these problems, scientists are developing schemes based on traditional digital signatures to help users verify the integrity of their data without having to retrieve them, which they term as provable data possession (PDP) or proofs of retrievability (POR). In this survey paper, we will provide an analysis to some latest research on this problem, as well as providing a look into the future, to eventually provide a big picture for this research topic.

The rest of this paper is organized as follows. Section 2 gives some motivating examples regarding security and privacy in big data application and cloud computing. Section 3 analyzes the research problem and propose a lifecycle of integrity verification over big data in cloud computing. Section 4 shows some representative approaches and their analyses. Section 5 provides a brief overview of other schemes in the field. Section 6 provides conclusions and points out future work.

For the convenience of readers, we list frequently-used acronyms in Table 1.

Table 1
Acronyms/abbreviations.

AAI	Auxiliary Authentication Information
ADS	Authenticated Data Structure
BLS	Boneh–Lynn–Shacham signature scheme
CSS	Cloud Storage Server
HLA	Homomorphic Linear Authenticator
HVT	Homomorphic Verifiable Tag
MAC	Message Authentication Code
MHT	Merkle Hash Tree
PDP	Provable Data Possession
POR	Proof of Retrievability
RASL	Rank-based Authenticated Skip List
TPA	Third-Party Auditor

2. Research motivations

Big data and cloud computing are currently receiving more and more attention from both industry and academia. They have been recently listed as important strategies by Australian Government [18,19]. To address big data problems, cloud computing is believed to be the most potent platform. In Australia, big companies such as Vodafone Mobile and News Corporation are already moving their business data and its processing tasks to Amazon cloud—Amazon Web Services (AWS) [20]. Email systems of many Australian universities are using public clouds as the backbone. To tackle the large amount of data in scientific applications, CERN, for example, is already putting the processing on petabytes of data into cloud computing [21]. There has also been a lot of research regarding scientific cloud computing, such as in [22–24]. For big data applications within cloud computing, data security is a problem that should always be properly addressed. In fact, data security is one of the biggest reasons why people are reluctant in using cloud [12,25,26]. Therefore, more effective and efficient security mechanisms are direly in need to help people establish their confidence in all-round cloud usage.

Data integrity is always an important part in data security, and there is no exception for cloud data [27]. As data in most big data applications are dynamic in nature, we will focus on verification of dynamic data. A large proportion of the updates are very small but very frequent. For example, a large proportion of data collected in the Internet of Things are numerical sensor data that are very small and dynamic, such as temperature and humidity data from the environment, or heart rate and blood pressure data from human body. Another example is social network data. In 2010 Twitter is producing up to 12 terabytes of data every day; this data is composed of tweets with size of 140 characters maximum [28]. Business transactions and loggings are also good examples. The dataset in these big data applications are very large in size and requires heavy-scale processing capabilities. Therefore, the requirements reside not only in security, but also in efficiency.

3. Problem analysis—framework and lifecycle

As stated in Section 1, external verification is as important as, if not more important than, server-side protection and internal verification. In this paper, we will focus on integrity protection and verification from external parties. There are 3 participating

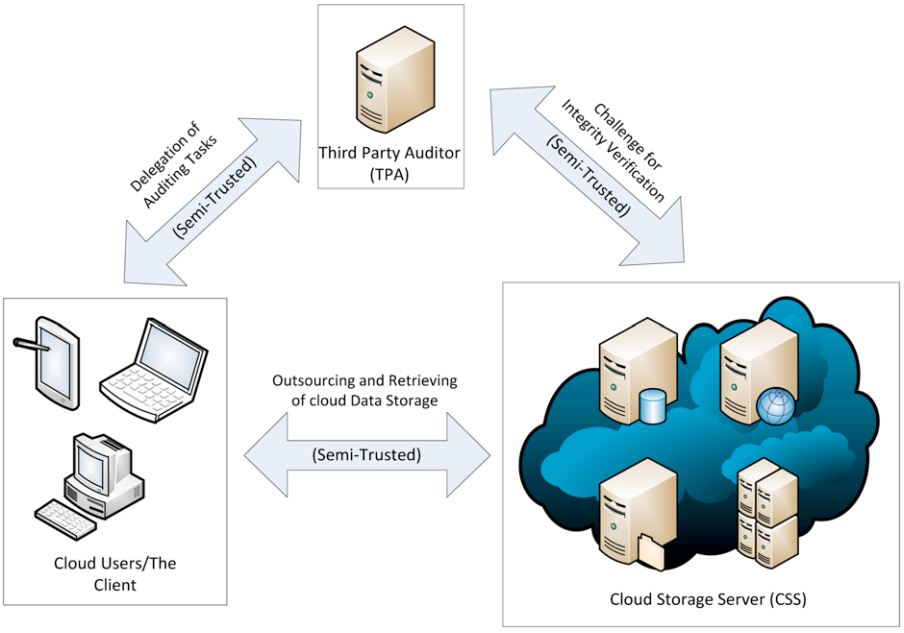

Fig. 1. Relations between the participating parties in external verification.

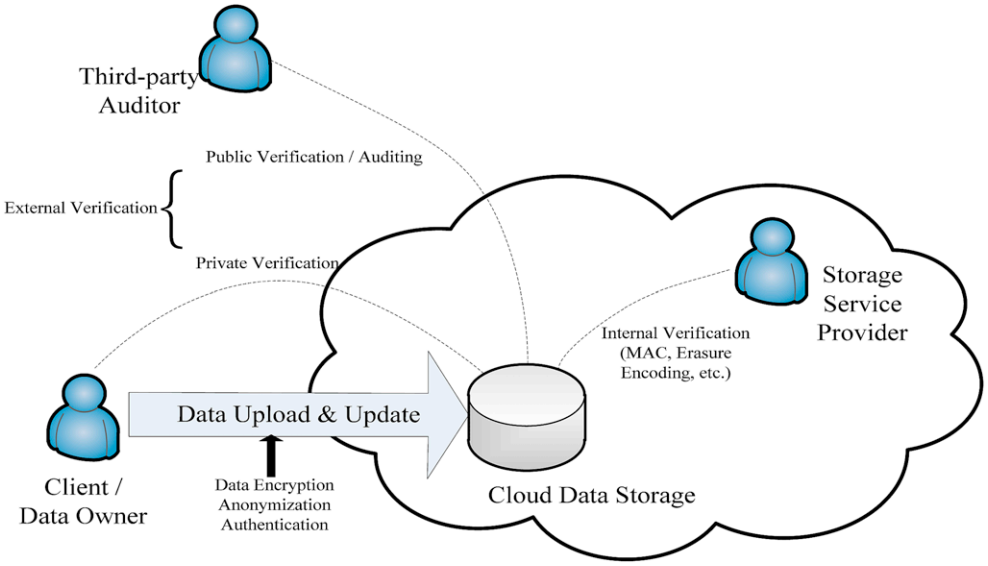

Fig. 2. Integrity verification for outsourced data—a framework.

parties in an integrity verification game: client, CSS and TPA. The client stores her data on CSS, while TPA's objective is to verify the integrity of client's data stored on CSS. Having a specialized TPA to verify data integrity is more efficient, but it may also introduce additional risk as the third-party auditor may not be fully trustworthy by itself, see Fig. 1. This has also been a widely concerned research problem over recent years. A framework of integrity protection on cloud data is shown in Fig. 2.

The main lifecycle of a remote integrity verification scheme (with support for dynamic data updates) can be analyzed in the following steps: Setup and data upload; Authorization for TPA; Challenge for integrity proof; Proof integration; Proof verification; Updated data upload; Updated metadata upload; Verification of updated data. The relationship and order of these steps are illustrated in Fig. 3. We now analyze in detail how these steps work and why they are essential to integrity verification of cloud data storage.

Setup and data upload: In cloud, user data is stored remotely on CSS. In order to verify the data without retrieving them, the client will need to prepare verification metadata, namely homomorphic linear authenticator (HLA) or homomorphic verifiable tag (HVT), based on homomorphic signatures [29]. Then, these metadata will be uploaded and stored alongside the original datasets. These tags are computed from the original data; they must be small in size in comparison to the original dataset for practical use.

Authorization for TPA: This step is not required in a two-party scenario where clients verify their data for themselves, but it is important when users require a semi-trusted TPA to verify the data on their behalf. If a third party can infinitely ask for integrity proofs over a certain piece of data, there will always be security risks in existence such as plain text extraction.

Challenge and verification of data storage: This step is where the main requirement – integrity verification – has to be fulfilled. The client will send a challenge message to the server, and server

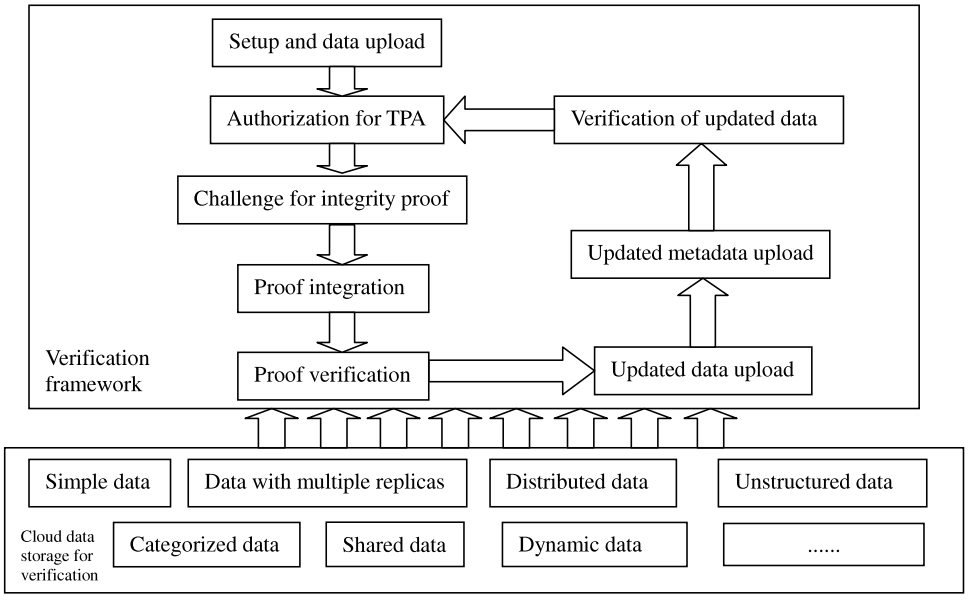

Fig. 3. Integrity verification over big data in cloud computing—lifecycle and research topics.

will compute a response over the pre-stored data (HLA) and the challenge message. This response is computed based on all message blocks, which we call 'proof integration' in the cycle. The client can then verify the response to find out whether the data is intact. The scheme has public verifiability if this verification can be done without the client's secret key. If the data storage is static, the whole process would have been ended here. However, as discussed earlier, data are always dynamic in many big data contexts (often denoted as velocity, one of the four V's). In these scenarios, we will need the rest of the steps to complete the lifecycle.

Data update: Occurs in dynamic data contexts. The client needs to perform updates to some of the cloud data storage. The updates could be roughly categorized in insertion, deletion and modification; if the data is stored in blocks with varied size for efficiency reasons, there will be more types of updates to address.

Metadata update: In order to keep the data storage stay verifiable without retrieving all the data stored and/or re-running the entire setup phase, the client will need to update the verification metadata (HLA or HVT's) with the existing keys.

Verification of updated data: This is also an essential step in dynamic data context. As the CSS is not completely trusted, the client needs to verify the data update process to see if the updating of both user data and verification metadata has been performed successfully in order to ensure that the updated data can still be verified correctly in the future.

We will show in the next section on how each step in this lifecycle was developed and evolved, through analyzing representative approaches in this research area.

4. Representative approaches and analysis

Research in integrity verification of remote data storage can be split into POR (proof of retrievability) and PDP (provable data possession). The basic idea behind the two models are the same: the file is separated into blocks and each block is accompanied by a small piece of metadata for verification. The main difference between the two is the security model: PDP aims at verification of integrity of most of the blocks through verifying a small number of blocks, while POR aims at verification of all data blocks through storage of redundantly encoded client data [30]. In this paper, we will mainly focus on works on the PDP setting as they are more efficient and practical to use.

We now introduce the common setting and some common notations for elaboration. The file m is stored in the form of a number of blocks, denoted as m_i. Each of the block is accompanied with a tag called HLA/HVT denoted as T_i, computed with the client's secret key. Therefore CSS cannot compute T_i (also frequently denoted as σ_i) from m_i. The client will choose a random set of m_i, send over the coordinates, and ask for proofs. CSS will compute a proof based on the tags T_i according to m_i. Due to homomorphism of the tags, the client will still be able to verify the proof with the same private key used for tag computation.

4.1. Preliminaries

We now introduce some preliminaries laid as foundation stones for our research area. HLA or HVT is evolved from digital signatures; current methods in verifiable updates utilized authenticated data structures. Therefore, we will introduce here two standard signature schemes (RSA and BLS) and one authenticated data structure (MHT) involved in representative approaches.

4.1.1. RSA signature

The RSA signature is a classic and one of the earliest signature schemes; it is also one of the foundation stones of public-key cryptography. While the textbook version is not semantically secure and not resilient to existential forgery attacks, there is a large body of research work on its improvements later on, and eventually makes it a robust signature scheme. For example, a basic improvement is to use $h(m)$ instead of m where h is a one-way hash function.

The setup is based on an integer $N = pq$ where p and q are two large primes, and two integers d and e where $ed = 1 \bmod N$; d is kept as the secret key and e is the public key. The signature σ of a message m is computed as $\sigma = m^d \bmod N$. Along with m, the signature can be verified through verifying whether the equation $m = \sigma^e \bmod N$ holds.

4.1.2. Bilinear pairing and BLS signature

BLS signature is proposed by Boneh, Lynn and Shacham [31] in 2004. In addition to the basic soundness of digital signature, this scheme has a greatly reduced signature length, but also increased overheads due to the computationally expensive paring operations.

Assume a group G is a gap Diffie–Hellman (GDH) group with prime order p. A bilinear map is a map constructed as $e : G \times G \rightarrow$

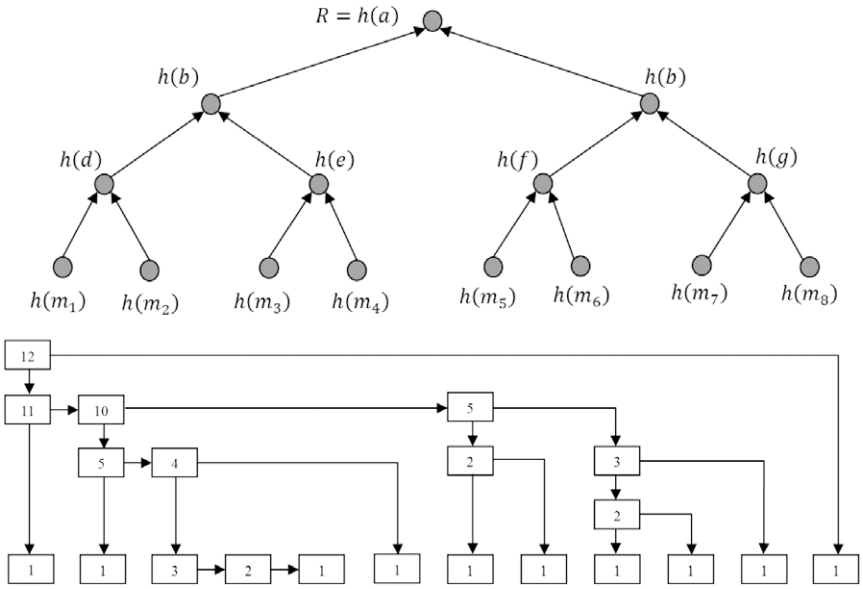

Fig. 4. ADS examples: MHT and RASL.

G_T, where G_T is a multiplicative cyclic group with prime order.[1] A usable e should have the following properties: bilinearity—$\forall m,\ n \epsilon G \Rightarrow e\left(m^a, n^b\right) = e(m, n)^{ab}$; non-degeneracy—$\forall m \in G,\ m \neq 0 \Rightarrow e(m, m) \neq 1$; and computability—$e$ should be efficiently computable. For simplicity, we will use this symmetric bilinear map in our scheme description. Alternatively, the more efficient asymmetric bilinear map in the form of $e : G_1 \times G_2 \rightarrow G_T$ may also be applied, as was pointed out in [31].

Based on a bilinear map $e : G \times G \rightarrow G_T$, a basic BLS signature scheme works as follows. Keys are computed as $y = g^x$ where $g \in G$, x is secret key and $\{g, y\}$ is public key. Signature σ for a message m is computed as $\sigma = (h(m))^x$. People can then verify this signature through verifying whether $e(\sigma, g) = e(h(m), y)$.

4.1.3. Authenticated data structures

Authenticated data structures (ADS) are used to efficiently verify data position through verification of all data in the verification path from the root. It is employed in integrity verification schemes to enable the verifier to check whether the storage server has performed data update correctly. Now we briefly introduce ADS used in integrity verification.

Merkle Hash Tree (MHT) [32] is an authenticated data structure which has been intensively studied in the past and later utilized to support verification of dynamic data updates. Similar to a binary tree, each node N will have a maximum of 2 child nodes. Information contained in one node N in an MHT T is $\mathcal{H}$—a hash value. T is constructed as follows. For a leaf node LN based on a message m_i, we have $\mathcal{H} = h(m_i)$, $r_{LN} = s_i$; A parent node of $N_1 = \{\mathcal{H}_1, r_{N1}\}$ and $N_2 = \{\mathcal{H}_2, r_{N2}\}$ is constructed as $N_P = \{h(\mathcal{H}_1 \parallel \mathcal{H}_2)\}$ where $\mathcal{H}_1$ and $\mathcal{H}_2$ are information contained in N_1 and N_2 respectively. A leaf node m_i's AAI Ω_i is defined as a set of hash values chosen from its upper levels (one hash per level) so that the root value R can be computed through $\{m_i, \Omega_i\}$. For example, for the MHT demonstrated in Fig. 4, m_1's AAI $\Omega_1 = \{h(m_2), h(e), h(b)\}$.

Rank-based authenticated skip list (RASL) [33] is an authenticated data structure that can authenticate not only the content, but also the indices of the data block. Based on this structure, Erway et al. proposed the first PDP scheme that can support full dynamic data operations. An example can also be found in Fig. 4, where the 'rank' value of a node is defined as the maximum number of leaf nodes it can reach. Its average complexity is also logarithmic to the number of blocks, similar to MHT.

There are also other authenticated data structures. Mo et al. designed Merkle B+ tree [34] which also has good complexity for updates. Liu et al.. have proposed ranked Merkle hash tree (RMHT) for fine-grained updates. However, the rank value is not for authentication of indices, but for authentication of variable block sizes.

4.2. Representative schemes

Now we start to introduce and analyze some representative schemes. These schemes are basically presented in a chronicle order, and the scheme presented later can support improved properties which we will analyze by the end of this section. Note that all computations are within the cyclic group $\mathbb{Z}_p$ or $\mathbb{Z}_N$.

4.2.1. PDP

Proposed by Ateniese et al. in 2007, PDP (provable data possession) can provide authors with efficient verification over their outsourced data storage [35,36]. It is the first scheme to provide blockless verification and public verifiability at the same time.

The tag construction is based on RSA signature, therefore all computations are modulo N by default. Let N, e, d be defined as the same as in RSA signature, g is a generator of QR_N, and v is a random secret value; $\{N, g\}$ is the public key and $\{d, v\}$ is the secret key. The tag is computed as $\sigma_i = (h(v \parallel i) \cdot g^{m_i})^d$. To challenge CSS, the client sends the indices (or, coordinates) of the blocks they want to verify, and correspondingly chooses a set of coefficients a_i, as well as a $g_s = g^s \mod N$ where s is a random number, and send them to CSS along with the indices. To prove data integrity, CSS will compute $\sigma = \prod_i \sigma_i^{a_i}$, along with a value $\rho = H(\prod_i g_s^{a_i m_i})$, and send back $\{\sigma, \rho\}$ as the proof. To verify this proof, the client (or TPA) will compute $\tau = \frac{\sigma^e}{\prod_i h(v \| i)^{a_i}}$, then verify if $\rho = H(\tau^s \mod N)$.

The authors have also proposed a light version called E-PDP, in contrast to the formal S-PDP scheme, for better efficiency. The basic idea is to throw away the coefficients a_i and compute the proof

[1] For simplicity, we only discuss symmetric pairing here, although specific asymmetric parings could also be applied for better efficiency.

ρ as $\rho = H(\prod_i g_s^{m_i})$, and the verification equation is therefore $\tau = \frac{\sigma^e}{\prod_i h(v\|i)}$. However, the light version was later proved not secure under the compact POR model. However, as a milestone in this research area, a lot of settings continued to be used by the following work. Mixing in random coefficients is one of the example. Another example is that the paper proposed a probability analysis and found that only a constant small number of blocks are to be verified, if the client needs to have 95% or even 99% confidence in that the integrity of the entire file is good. This analysis also became a default setting in the following schemes.

4.2.2. Compact POR

Compact POR is proposed by Shacham et al. in 2008 [37]. Compared to original POR, the authors provided an improved rigorous security proof. The schemes they introduced in the paper also suit the PDP model.

They proposed first a construction for private verification. In this case, data can only be verified with the secret key, therefore no other party can verify it except for the client. The metadata HVT is computed as $\sigma_i = f_k(i) + \alpha m_i$, where $f_k()$ is a pseudo-random function (PRF). α and the PRF key k is kept as a secret key. When the server is challenged with a set of block coordinates and a set of corresponding public coefficients v_i (same definition as a_i in PDP above), it will compute $\sigma = \sum_i v_i\sigma_i$ and $\mu = \sum_i v_i m_i$ to return $\{\sigma, \mu\}$ as the proof. Upon receiving the proof, the client can simply verify if $\sigma = \alpha\mu + \sum_i (v_i f_k(i))$. The scheme is efficient because it admits short response length and fast computation.

The other construction with public verification is even more impressive compared to schemes at that time. It is the first BLS-based scheme that supports public verification. Due to the shortened length of BLS signature, the proof size is also greatly reduced compared to RSA-based schemes. Similar to BLS signature, the tag construction is based on a bilinear map $e : G \times G \to G_T$ where G is a group of prime order p. Two generators g and u of $\mathbb{Z}_p$ are chosen to be the public key, as another value $v = g^\alpha$ where α is the secret key for the client. The tag is computed as $\sigma_i = (H(i)\, u^{m_i})^\alpha$, Same as the one with private verification, a set of coefficients v_i is also chosen with the designated block coordinates. When challenged, the proof $\{\sigma, \mu\}$ is computed as $\sigma = \prod_i \sigma_i^{v_i}$ and $\mu = \sum_i v_i m_i$. The client can then verify the data integrity through verifying if $e(\sigma, g) = e\left(\prod_i (H(i)^{v_i}) \cdot u^\mu,\ v\right)$.

Another great contribution of this work is the rigorous security framework it provided. In their model, a verification scheme is secure only when it is secure against an arbitrary adversary with a polynomial extraction algorithm to reveal the message from the integrity proof. To prove the security, they also defined a series of interactive games under the random oracle model. Compared to the previous security frameworks in first PDP and first POR schemes, the adversary defined in this framework is stronger and stateless, and the definition of extraction algorithm (therefore the overall soundness) is stronger. Also, their framework suits perfectly with the public verification, and even multi-replica storage and multi-prover scenarios. To date, this model is considered the strongest and is very frequently used to prove the security of newly-proposed verification algorithms.

4.2.3. DPDP

DPDP (Dynamic PDP), proposed in 2009, is the first integrity verification scheme to support full data dynamics [33]. It is from here that the processes in integrity verification schemes started to form a self-closed lifecycle. They utilized another authenticated data structure – rank-based authenticated skip list – for verification of updates. A rank-based skip list is similar to MHT in the sense that they will both incur a logarithm amount of operations when an update occurs. All types of updates – insert, delete and modification – are supported for the first time. This design is essentially carried on by all the following schemes with dynamic data support. However, public verifiability was not supported by the scheme, and there was no follow-up work to fill in the blank. Therefore, we will only give a brief introduction here. The readers can refer to the next subsection to see how data dynamics is supported with an authenticated data structure such as MHT.

4.2.4. Public auditing of dynamic data

As the DPDP scheme did not provide support for public verifiability, Wang et al. proposed a new scheme that can support both dynamic data and public verifiability at the same time [38]. They term the latter as 'public auditability', as the verification is often done by a sole-duty third-party auditor (TPA).

An MHT is utilized to verify the updates where the root R is critical authentication information. The tree structure is constructed on blocks, and the structure is stored along with the verification metadata. Compared to compact POR, they compute the tags using $H(m_i)$ instead of $H(i)$ in order to support dynamic data, otherwise all tags of the following blocks must be changed upon each one insert or delete update, which will be very inefficient. Aside from this, the tag construction and verification are similar: $\sigma_i = (H(m_i)\, u^{m_i})^\alpha$. The proof is also computed as $\sigma = \prod_i \sigma_i^{v_i}$ $\mu = \sum_i v_i m_i$. While the verification is to verify whether $e(\sigma, g) = e\left(\prod_i (H(m_i)^{v_i}) \cdot u^\mu,\ v\right)$, TPA will first verify $H(R)$'s signature to ensure that the MHT is correct at server side.

To verify data updates, the client will first generate the tag for new block: $\sigma_i' = \left(H(m_i')\, u^{m_i'}\right)^\alpha$, then upload it to CSS along with the update request. CSS will update the metadata as requested, and send back R' along with the old block $H(m_i)$, the AAI Ω_i (note Ω_i will stay unchanged if m_i is the only block that has changed) and the client-signed old MHT root $H(R)$. The client can then verify the signed $H(R)$ to ensure that CSS has not manipulated it, then it can verify R' with m_i' and Ω_i to see if the update of data and metadata was correct. Except for the main scheme, they also proposed a scheme that can perform efficient batch auditing with experimental results.

There was also a follow-up work to improve this scheme for privacy preserving public auditing [39]. When computing integrity proof, they added a random masking technique to prevent the part of original file being extracted from several integrity proofs over this specific part of data.

4.2.5. Authorized auditing with fine-grained data updates

Although the above schemes have already supported dynamic data and public verification/auditability, they only support insert/delete/modification with blocks with a fixed size, which are later termed as 'coarse-grained updates'. Lack of support of fine-grained updates, i.e., arbitrary-length updates, especially small updates, will cause functionality and efficiency problems. Liu et al. [40] proposed a public auditing scheme with the support of fine-grained updates over variable-sized file blocks. In addition, an authentication process between the client and TPA is also proposed to prevent TPA from endless challenges, thereby cut the possibility of attacks over multiple challenges (like the one in [39]) from source.

Similar to previous work, this scheme is also based on BLS signature. Unlike previous schemes which are based on evenly distributed file blocks, here the file blocks are of variable size, with an upper bound of $s_{\max}$ sectors per block. The tag construction is $\sigma_i = \left(H(m_i) \prod_{j=1}^{s_i} u_j^{m_{ij}}\right)^\alpha$ where $u_j \in U$, $U = \left\{u_k \in \mathbb{Z}_p\right\}$, $k \in [1, s_{\max}]$ is chosen according to $s_{\max}$. To challenge CSS, TPA must first obtain authorization from client to be eligible for auditing. The client will compute $sig_{AUTH} = Sig_{ssk}(AUTH\|t\|VID)$, which is a signature with client's secret key where *VID* is the verifier ID and *AUTH* is a message shared secretly earlier between client and CSS.

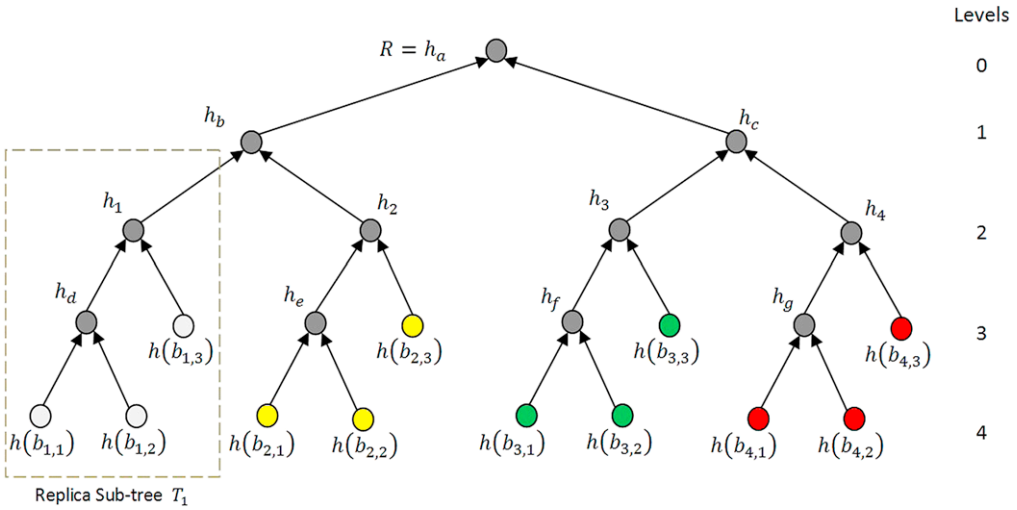

Fig. 5. Multi-replica Merkle Hash Tree.

In this case, only the client can generate this signature and only the CSS (other than the client herself) will be able to verify sig_{AUTH}. After CSS has finished verifying sig_{AUTH}, it will compute the proof $P = \{\sigma, \{\mu_k\}_{k\in[1,w]}, \{H(m_i), \Omega_i\}_{i\in I}, sig\}$ where $\sigma = \prod_i \sigma_i^{v_i}$ and $\mu_k = \sum_{i\in I} v_i m_{ik}$, then send P back to TPA. TPA will then verify the proof through verifying whether $e(sig, g) = e(H(R), v)$ and $e(\sigma, g) = e(\omega, v)$, where $\omega = \prod_{i\in I} H(m_i)^{v_i} \cdot \prod_{k\in[1,w]} u_k^{\mu_k}$.

For support in fine-grained updates, 5 types of necessary updating operations including $\mathcal{PM}$, $\mathcal{M}$, $\mathcal{D}$, $\mathcal{I}$ and $\mathcal{SP}$ are analyzed; a theorem was provided to illustrate that all updates can be divided into these 5 basic operations. For more efficient verification of fine-grained updates, a modified verification scheme for $\mathcal{PM}$ operations (which was the majority of the operations in many occasions found through analysis) is also provided, where only the modified part of the new block, instead of a whole block, is needed to be retrieved and transferred back to the client for tag re-computation. Experimental results have also demonstrated some significant efficiency improvements.

4.2.6. MuR-DPA: support for verification multiple replicas and index verification

Storing multiple replicas is a common strategy for reliability and availability in cloud. For highly dynamic data, each update will lead to update to every replica. Given the fact that update verifications in current verification schemes are of $O(\log n)$ communication complexity, verifying these replicas one by one will be very costly in terms of communication. Also, current schemes for dynamic public auditing are susceptible to attacks from dishonest servers because of lack of index verification. Although there is an integrity verification scheme for dataset with replicas [41] and schemes with index verification such as [33], there will be security and/or efficiency problems if these schemes are extended directly to support public verifiability. Liu, et al. proposed a scheme named MuR-PDP [42] that can support efficient public auditing of multiple replicas of dynamic datasets with authentication of block indices. To address these problems, a new ADS called MR-MHT (see Fig. 5) is proposed, where all replicas of a certain block are organized into one sub-tree, and the hash values in MR-MHT are computed with additional block index information. For support of public auditing and dynamic data, the level information is generated top-down, i.e., the level for root value is 0, level for child nodes of root is 1, etc. Different to RASL, the indices of leaf nodes are verified through computation of rank information from both left to right and right to left. Experimental results show that the scheme has significant efficiency advantage for auditing multi-replica cloud storage of dynamic data, while achieving a better security level.

A qualitative comparison of the schemes described in this section is provided in Table 2, where it can be seen that the properties (goals) are various and very different. Therefore, an overall quantitative comparison will be tedious and unnecessary. Interested readers may infer to the original papers for detailed quantitative results.

5. Other related work

Other than the ones stated in the previous section, a great amount of work has also been proposed in recent years to address the research problem of integrity verification and public auditing of cloud data and other outsourced data storage. The concept of POR is proposed in 2007 by Juels et al. [43], but the security framework was not complete and it only suits for static data storage like library and archives. After PDP, Ateniese et al. also proposed an improvement they call Scalable PDP [44] to support dynamic data verification. Alas, only partial data dynamics is supported, i.e., only limited types of data updates is supported. Therefore, this scheme is not suitable for practical use. Curtmola et al. proposed a verification for multi-replica cloud storage, which is named MR-PDP [41]. This is also a practical solution, because most cloud deployments now constantly keep a number of replicas of user data in the aim of availability. Ateniese et al. also proposed a framework to transfer homomorphic identification protocols into integrity verification schemes [45].

There are also some works proposed in the most recent years. Based on previous work and the recent developments of big data and cloud, they can be more practical solutions for specific cloud environments and applications. As mentioned before, [40] is a good example. For big enterprises, data migration is a big problem in the adoption process of cloud, due to the different security/control levels in data and the heavy cost in migration itself. Therefore, hybrid cloud has been a more practical solution; enterprises will keep relatively static and security-sensitive data on private cloud, and put all services into the cloud. Zhu et al. proposed a PDP scheme for Hybrid Cloud [11] for verification of data stored in separated domains. As cloud data sharing is happening in all, Wang et al. worked on secure data verification of shared data storage [46] and also with efficient user management [47] and user privacy protection [48]. Zhang et al. proposed a scheme with a new data structure called update tree [49]. Without conventional authenticated data structures such as MHT, the proposed scheme

Table 2
Comparison of external integrity verification schemes.

	POR [14]	PDP [10]	Scalable PDP [8]	Compact POR [37]	MR-PDP [41]	DPDP [33]	Dynamic public auditing [38]	FU-DPA [40]	MuR-DPA [42]
Blockless verification	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Stateless verification	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Infinite verifications	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Public verifiability	No	Yes	No	Yes	No	No	Yes	Yes	Yes
Coarse-grained verifiable data updating	No	No	Partly	No	No	Yes	Yes	Yes	Yes
Fine-grained verifiable data updating	No	No	No	No	No	No	No	Yes	Capable
Variable-sized data blocks	No	Yes	Yes	No	Yes	Yes	No	Yes	Yes
Authorized auditing	No	No	No	No	No	No	No	Yes	Yes
Authentication of node indices (for schemes with ADS)	N/A	N/A	N/A	N/A	N/A	Yes	No	No	Yes
One interaction for updating all replicas	No	No	No	No	No	No	No	No	Yes

has a constant proof size and support fully data dynamics. However, the scheme does not support public verification (public auditing) at the moment. Cash et al. [50] proposed a novel POR scheme based on oblivious RAM (ORAM). ORAM, or oblivious file system, was mostly used to hide data access patterns through shuffling and noise addition on outsourced data storage [51,52]. Shi et al. also proposed a more practical and efficient scheme based on ORAM [53]. These schemes also support public verifiability and dynamic data storage, with the same level of complexity with MHT/RASL based schemes.

6. Conclusions and future work

As we can see from the above, the topic of integrity verification of big data in cloud computing is a flourishing area that is attracting more and more research interest and there is still lots of research currently ongoing in this area. Cloud and big data is a fast-developing topic. Therefore, even though existing research has already achieved some amazing goals, we are confident that integrity verification mechanisms will also continue evolving along with the development of cloud and big data applications to meet emerging new requirements and address new security challenges. For future developments, the following aspects are particularly interesting to look at.

Efficiency: Due to high efficiency demands in big data processing overall, efficiency is one of the most important factors in designing of new techniques related to big data and cloud. In integrity verification/data auditing, the main costs can come from every aspect, including storage, computation and communication, and they can all affect the total cost-efficiency due to the pay-as-you-go model in cloud computing. We now analyze these three aspects one by one for a scheme with public auditability and support of full dynamic verifiable data updates.

(a) Communication and storage: These two are the main efficiency concerns of public auditing schemes. One of the most challenging problems is that due to usage of ADS, the size of proofs depends logarithmically on the total size of the dataset, which constitutes the main communication overhead for verification of updates. Similarly, the authenticators take extra storage overhead at server side, which also grows with the growth of total size of datasets. Although there are works for their optimizations, the ideal case is that the proof size and storage overhead remains constant. To the best of our knowledge, these desirable properties have never been achieved by any dynamic public auditing scheme.
(b) Computation time: It is not the primary concern, but also important. The computation time for proof generation can be considered negligible in most cases, but the pre-processing time can sometimes be considerable for incremental datasets.

Security: Security is always a problem between spear and shield; that is, attack and defense. Although the current formalizations and security model seemed very rigorous and potent, new exploits can always exist, especially with dynamic data streams and varying user groups. Finding the security holes and fixing them can be a long-lasting game. The security focuses of existing work can be summarized by different roles of adversaries: dishonest cloud servers [33,42], malicious TPA [40], other malicious users [47], and other general-sense attackers [7,54]. With the proposed authentication mechanisms in [33,42], exploits from dishonest servers can be effectively detected in data updates. Based on existing research, a most attractive future research topic will be letting the TPA to get minimized information on client data during auditing. There may also be big potential in addressing security threats from other malicious users. Multi-tenancy is one of the cloud's main characteristics, and there is currently not much work focusing on investigating this area.

Scalability/elasticity: As the cloud is a parallel distributed computing system in nature, scalability is one of the key factors as well. Programming models for parallel and distributed systems, such as MapReduce, are attracting attentions from a great number of cloud computing researchers. Some of the latest work in integrity verification is already considering how to work well with MapReduce for better parallel processing [11]. On the other hand, elasticity is one of a biggest reason why big companies are moving their business, especially service-related businesses, to the cloud [20]. User demands vary all the time, and it would be a waste of money to purchase hardware that can handle the demands at peak times. The advent of cloud solved this problem—cloud allows their clients to deploy their applications on a highly elastic platform whose capabilities can be scaled up and down on-the-fly, and the cost is based solely on usage. Therefore, an integrity verification mechanism that has the same level of scalability and elasticity will be highly resourceful for big data applications in cloud environments.

Acknowledgments

This research work is supported in part by Australia Research Council Linkage Project LP0990393 and CSIRO OCE Scholarship R-812-25.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (2010) 50–58.
[2] W. Dou, X. Zhang, J. Liu, J. Chen, Hiresome-II: towards privacy-aware cross-cloud service composition for big data applications, IEEE Trans. Parallel Distrib. Syst. (TPDS) (2013) (in press).
[3] F. Hao, G. Min, J. Chen, F. Wang, M. Lin, C. Luo, L.T. Yang, An optimized computational model for task-oriented multi-community-cloud social collaboration, IEEE Trans. Serv. Comput. (TSC) (2014) (in press).
[4] X. Zhang, L.T. Yang, C. Liu, J. Chen, A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud, IEEE Trans. Parallel Distrib. Syst. (TPDS) 25 (2014) 363–373.

[5] X. Zhang, C. Liu, S. Nepal, C. Yang, W. Dou, J. Chen, A hybrid approach for scalable sub-tree anonymization over big data using MapReduce on cloud, J. Comput. System Sci. (JCSS) 80 (2014) 1008–1020.
[6] S. Meng, W. Dou, X. Zhang, J. Chen, Kasr: a keyword-aware service recommendation method on mapreduce for big data application, IEEE Trans. Parallel Distrib. Syst. (TPDS) (2013) (in press).
[7] C. Liu, X. Zhang, C. Yang, J. Chen, CCBKE—session key negotiation for fast and secure scheduling of scientific applications in cloud computing, Future Gener. Comput. Syst. 29 (2013) 1300–1308.
[8] C. Yang, C. Liu, X. Zhang, S. Nepal, J. Chen, A time efficient approach for detecting errors in big sensor data on cloud, IEEE Trans. Parallel Distrib. Syst. (2013) (in press).
[9] C. Yang, X. Zhang, C. Zhong, C. Liu, J. Pei, K. Ramamohanarao, J. Chen, A spatiotemporal compression based approach for efficient big data processing on cloud, J. Comput. System Sci. (JCSS) 80 (2014) 1563–1583.
[10] K. Zhang, X. Zhou, Y. Chen, X. Wang, Y. Ruan, Sedic: privacy-aware data intensive computing on hybrid clouds, in: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS'11, 2011, pp. 515–526.
[11] Y. Zhu, H. Hu, G.-J. Ahn, M. Yu, Cooperative provable data possession for integrity verification in multi-cloud storage, IEEE Trans. Parallel Distrib. Syst. 23 (2012) 2231–2244.
[12] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Gener. Comput. Syst. 28 (2011) 583–592.
[13] X. Zhang, C. Liu, S. Nepal, J. Chen, An efficient quasi-identifier index based approach for privacy preservation over incremental data sets on cloud, J. Comput. System Sci. (JCSS) 79 (2013) 542–555.
[14] X. Zhang, C. Liu, S. Nepal, S. Panley, J. Chen, A privacy leakage upper-bound constraint based approach for cost-effective privacy preserving of intermediate datasets in cloud, IEEE Trans. Parallel Distrib. Syst. 24 (2013) 1192–1202.
[15] L. Atzoria, A. Ierab, G. Morabito, The Internet of things: a survey, Comput. Netw. 54 (2010) 2787–2805.
[16] G. Kortuem, F. Kawsar, V. Sundramoorthy, D. Fitton, Smart objects as building blocks for the Internet of things, IEEE Internet Comput. 14 (2010) 44–51.
[17] Y. Ma, J. Rao, W. Hu, X. Meng, X. Han, Y. Zhang, Y. Chai, C. Liu, An efficient index for massive IoT data in cloud environment, in: Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM'12, Hawaii, USA, 2012, pp. 2129–2133.
[18] Australia Government, Department of Finance and Deregulation, Big Data Strategy—Issues Paper 2013. Available: http://agimo.gov.au/files/2013/03/Big-Data-Strategy-Issues-Paper1.pdf (accessed on 5.06.2014).
[19] Australia Government, Department of Finance and Deregulation, Cloud computing strategic direction paper: opportunities and applicability for use by the Australian Government 2011. Available: http://agimo.gov.au/files/2012/04/final_cloud_computing_strategy_version_1.pdf (accessed on 5.06.2014).
[20] Customer presentations on Amazon summit. Available: http://aws.amazon.com/apac/awssummit-au/ (accessed on 5.06.2014).
[21] N. Heath. Cern: Cloud computing joins hunt for origins of the universe. Available: http://www.techrepublic.com/blog/european-technology/cern-cloud-computing-joins-hunt-for-origins-of-the-universe/262 (accessed on 5.06.2014).
[22] C. Vecchiola, R.N. Calheiros, D. Karunamoorthy, R. Buyya, Deadline-driven provisioning of resources for scientific applications in hybrid clouds with Aneka, Future Gener. Comput. Syst. 28 (2012) 58–65.
[23] L. Wang, M. Kunze, J. Tao, G.v. Laszewski, Towards building a cloud for scientific applications, Adv. Eng. Softw. 42 (2011) 714–722.
[24] L. Wang, J. Tao, M. Kunze, A.C. Castellanos, D. Kramer, W. Karl, Scientific cloud computing: early definition and experience, in: Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications, HPCC'08, Dalian, China, 2008, pp. 825–830.
[25] S.E. Schmidt, Security and privacy in the AWS Cloud. Available: http://aws.amazon.com/apac/awssummit-au/ (accessed on 5.06.2014).
[26] J. Yao, S. Chen, S. Nepal, D. Levy, J. Zic, Truststore: making Amazon S3 trustworthy with services composition, in: Proceedings of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID'10, Melbourne, Australia, 2010, pp. 600–605.
[27] S. Nepal, S. Chen, J. Yao, D. Thilakanathan, Diaas: data integrity as a service in the cloud, in: Proceedings of the 4th International Conference on Cloud Computing, IEEE CLOUD'11, 2011, pp. 308–315.
[28] E. Naone, What Twitter learns from all those tweets. Available: http://www.technologyreview.com/view/420968/what-twitter-learns-from-all-those-tweets/ (accessed on 5.06.2014).
[29] R. Johnson, D. Molnar, D. Song, D. Wagner, Homomorphic signature schemes, in: Topics in Cryptology—CT-RSA 2002, in: Lecture Notes in Computer Science, vol. 2271, 2002, pp. 244–262.
[30] A. Küpçü, Efficient Cryptography for the Next Generation Secure Cloud: Protocols, Proofs, and Implementation, Lambert Academic Publishing, 2010.
[31] D. Boneh, H. Shacham, B. Lynn, Short signatures from the Weil pairing, J. Cryptology 17 (2004) 297–319.
[32] R.C. Merkle, A digital signature based on a conventional encryption function, in: Proceedings of A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology, CRYPTO'87, 1987, pp. 369–378.
[33] C. Erway, A. Küpçü, C. Papamanthou, R. Tamassia, Dynamic provable data possession, in: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS'09, Chicago, USA, 2009, pp. 213–222.
[34] Z. Mo, Y. Zhou, S. Chen, A dynamic proof of retrievability (Por) scheme with O(Logn) complexity, in: Proceedings of the 2012 IEEE International Conference on Communications, ICC'12, 2012, pp. 912–916.
[35] G. Ateniese, R.B. Johns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, Provable data possession at untrusted stores, in: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS'07, 2007, pp. 598–609.
[36] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, D. Song, Remote data checking using provable data possession, ACM Trans. Inf. Syst. Secur. 14 (2011) Article 12.
[37] H. Shacham, B. Waters, Compact proofs of retrievability, in: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'08, 2008, pp. 90–107.
[38] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, Enabling public auditability and data dynamics for storage security in cloud computing, IEEE Trans. Parallel Distrib. Syst. 22 (2011) 847–859.
[39] C. Wang, S.M. Chow, Q. Wang, K. Ren, W. Lou, Privacy-preserving public auditing for secure cloud storage, IEEE Trans. Comput. 62 (2013) 362–375.
[40] C. Liu, J. Chen, L.T. Yang, X. Zhang, C. Yang, R. Ranjan, K. Ramamohanarao, Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates, IEEE Trans. Parallel Distrib. Syst. (TPDS) 25 (2014) 2234–2244.
[41] R. Curtmola, O. Khan, R.C. Burns, G. Ateniese, MR-PDP: multiple-replica provable data possession, in: Proceedings of the 28th IEEE International Conference on Distributed Computing Systems, ICDCS'08, Beijing, China, 2008, pp. 411–420.
[42] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, J. Chen, MUR-DPA: top–down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud, IACR Cryptology ePrint Archive, Report 2014/391, 2014.
[43] A. Juels, J.B.S. Kaliski, PORs: proofs of retrievability for large files, in: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS'07, Alexandria, USA, 2007, pp. 584–597.
[44] G. Ateniese, R.D. Pietro, L.V. Mancini, G. Tsudik, Scalable and efficient provable data possession, in: Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, SecureComm'08, İstanbul, Turkey, 2008, pp. 1–10.
[45] G. Ateniese, S. Kamara, J. Katz, Proofs of storage from homomorphic identification protocols, in: Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT'09, Tokyo, Japan, 2009, pp. 319–333.
[46] B. Wang, S.S.M. Chow, M. Li, H. Li, Storing shared data on the cloud via security-mediator, in: 33rd IEEE International Conference on Distributed Computing Systems, ICDCS'13, Philadelphia, USA, 2013.
[47] B. Wang, B. Li, H. Li, Public auditing for shared data with efficient user revocation in the cloud, in: Proceedings of the 32nd Annual IEEE International Conference on Computer Communications, INFOCOM'13, Turin, Italy, 2013, pp. 2904–2912.
[48] B. Wang, B. Li, H. Li, Oruta: privacy-preserving public auditing for shared data in the cloud, in: Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, CLOUD'12, Hawaii, USA, 2012, pp. 295–302.
[49] Y. Zhang, M. Blanton, Efficient dynamic provable possession of remote data via update trees, IACR Cryptology ePrint Archive, Report 2012/291, 2012.
[50] D. Cash, A. Küpçü, D. Wichs, Dynamic proofs of retrievability via oblivious RAM, in: Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'13, Athens, Greece, 2013, pp. 279–295.
[51] E. Stefanov, M.v. Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, S. Devadas, Path Oram: an extremely simple oblivious RAM protocol, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS'13, 2013, pp. 299–310.
[52] P. Williams, R. Sion, A. Tomescu, Privatefs: a parallel oblivious file system, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS'12, 2012, pp. 977–988.
[53] E. Shi, E. Stefanov, C. Papamanthou, Practical dynamic proofs of retrievability, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS'13, 2013, pp. 325–336.
[54] C. Liu, X. Zhang, C. Liu, Y. Yang, R. Ranjan, D. Georgakopoulos, J. Chen, An iterative hierarchical key exchange scheme for secure scheduling of big data applications in cloud computing, in: Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, IEEE TrustCom'13, 2013, pp. 9–16.

Chang Liu is currently a Ph.D. student at the Faculty of Engineering and IT, University of Technology Sydney, Australia. Before joining UTS, he has received B.Eng. and M.Sc. degrees from Shandong University, China. His research interests include cloud and distributed computing, resource management, cryptography and data security.

Chi Yang received his B.S. degree from Shandong University At Wei-hai, China. He received his M.S. (by research) in Computer Science from Swinburne University of Technology, Melbourne, Australia, in 2007. Currently, he is a full-time Ph.D. student at the University of Technology Sydney, Australia. His major research interests include distributed computing, XML data stream, scientific workflow, Distributed System, Green Computing, Big Data Processing and Cloud Computing.

Xuyun Zhang is currently working towards the Ph.D. degree at the Faculty of Engineering and IT, University of Technology Sydney, Australia. Before joining UTS, he has received his Master's and Bachelor's degree in Computer Science from Nanjing University, China. His research interests include cloud computing, privacy and security, Big Data, MapReduce and OpenStack. He has published several papers in refereed international journals including IEEE Transactions on Parallel and Distributed Systems (TPDS).

Jinjun Chen is an Associate Professor from the Faculty of Engineering and IT, University of Technology Sydney (UTS), Australia. He is the Director of Lab of Cloud Computing and Distributed Systems at UTS. He holds a Ph.D. in Computer Science and Software Engineering from Swinburne University of Technology, Australia. His research interests include cloud computing, big data, workflow management, privacy and security, and related various research topics. His research results have been published in more than 100 papers in high quality journals and at conferences, including IEEE Transactions on Service Computing, ACM Transactions on Autonomous and Adaptive Systems, ACM Transactions on Software Engineering and Methodology (TOSEM), IEEE Transactions on Software Engineering (TSE), and IEEE Transactions on Parallel and Distributed Systems (TPDS). He received Swinburne Vice-Chancellor's Research Award for early career researchers (2008), IEEE Computer Society Outstanding Leadership Award (2008–2009) and (2010–2011), IEEE Computer Society Service Award (2007), Swinburne Faculty of ICT Research Thesis Excellence Award (2007). He is an Associate Editor for IEEE Transactions on Parallel and Distributed Systems. He is the Vice Chair of IEEE Computer Society's Technical Committee on Scalable Computing (TCSC), Vice Chair of Steering Committee of Australasian Symposium on Parallel and Distributed Computing, Founder and Coordinator of IEEE TCSC Technical Area on Workflow Management in Scalable Computing Environments, Founder and Steering Committee Co-chair of International Conference on Cloud and Green Computing, and International Conference on Big Data and Distributed Systems.