Public Auditing for Big Data Storage in Cloud Computing -- A Survey

Chang Liu*, Rajiv Ranjan+, Xuyun Zhang*, Chi Yang*, Dimitrios Georgakopoulos+, Jinjun Chen*
*Faculty of Engineering and IT, University of Technology Sydney, Australia
+Computational Informatics, CSIRO, Australia
{changliu.it, rranjans, xyzhanggz, chiyangit, jinjun.chen}@gmail.com, dimitrios.georgakopoulos@csiro.au

Abstract—**Data integrity is an important factor to ensure in almost any data and computation related context. It serves not only as 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 big data analytics, verification of data integrity becomes increasingly important, especially on outsourced data. Therefore, research topics related to data integrity verification have attracted tremendous research interest. Among all the metrics, efficiency and security are two of the most concerned measurements. In this paper, we provide an analysis on authenticator-based efficient data integrity verification. we will analyze and provide a survey on the main aspects of this research problem, summarize the research motivations, methodologies as well as main achievements of several of the representative approaches, then try to bring forth a blueprint for possible future developments.**

Keywords—cloud computing; big data; data security; integrity verification; public auditing

I. INTRODUCTION

Big data is attracting more and more interests from numerous industries. 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 in data center technology, parallel distributed systems and service computing, is widely considered as the most promising technological backbone for solving big data problems [2]. 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. 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.

Despite those stand-out advantages of cloud, there are still strong concerns regarding service qualities, especially data security. 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 computer science researchers' interest in recent years.

Main dimensions in data security include confidentiality, integrity and availability. In this paper, we will focus on data integrity. 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)

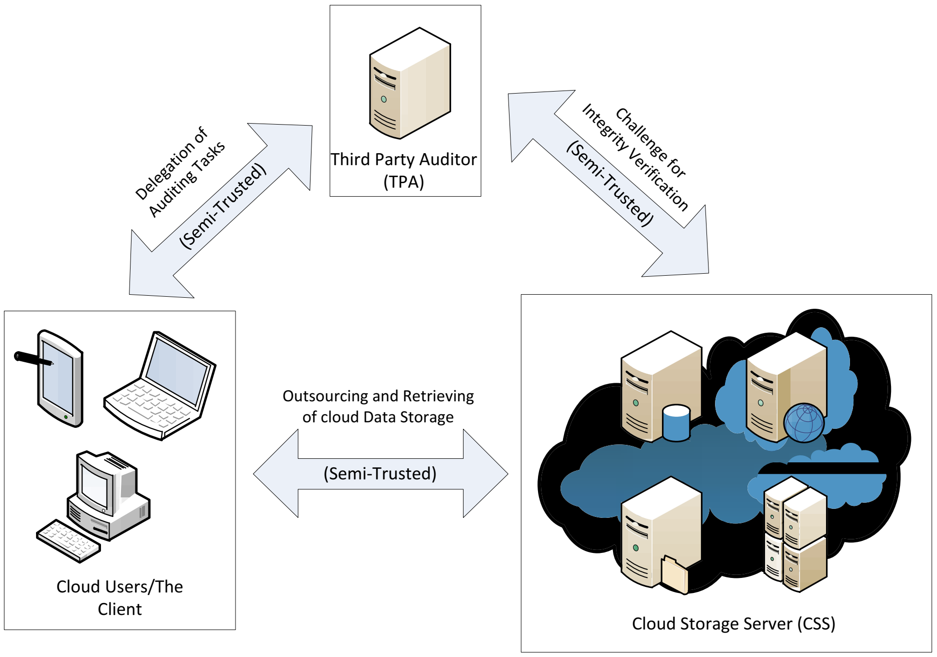
Third Party Auditor (TPA)
Delegation of Auditing Tasks
(Semi-Trusted)
Challenge for Integrity Verification
(Semi-Trusted)
Outsourcing and Retrieving of cloud Data Storage
(Semi-Trusted)
Cloud Users/The Client
Cloud Storage Server (CSS)

Fig 1: Relations between the participating parties

still needs a method to verify their data stored remotely on a semi-trusted cloud server, no matter how secure the cloud claim to be. 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. To address this problem, scientists are developing schemes mainly 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 survey 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.

AAI	Auxiliary Authentication Information
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 Retreivability
TPA	Third-Party Auditor

Table 1: Acronyms / abbreviations

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

II. Motivating Examples

Big data and cloud computing is receiving more and more interest from both industry and academia nowadays. They have been recently listed as important strategies by Australian Government [9, 10]. 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) [1]. Email systems of many Australian universities are using cloud as the backbone. To tackle the large amount of data in scientific applications, CERN, for example, is already putting the processing of petabytes of data into cloud computing [12]. There has also been a lot of research regarding scientific cloud computing, such as in [21, 26, 27]. 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 [19, 29, 32]. Therefore, more effective and efficient security mechanisms are direly in need to help people establish their confidence in all-around cloud usage.

Data integrity is always an important part in data security, and there is no exception for cloud data [18]. As stated in Section 1, client-side verification is as important as server-side protection. 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

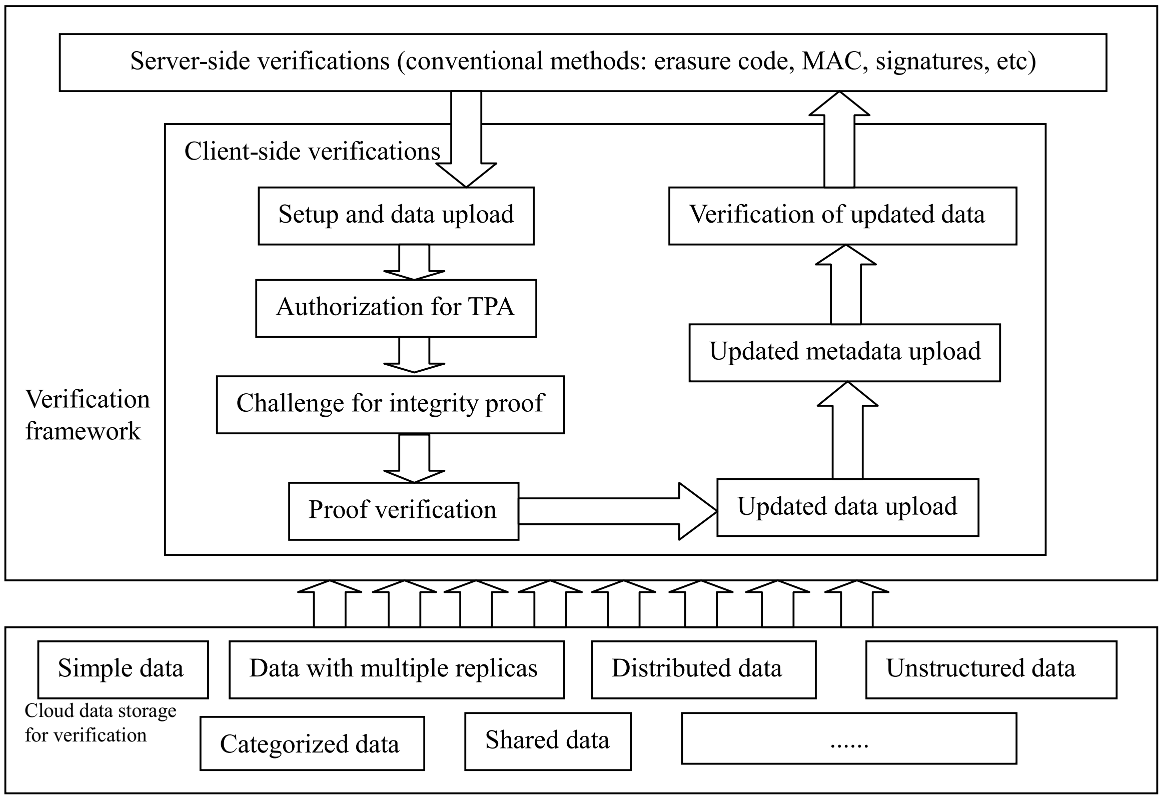
Server-side verifications (conventional methods: erasure code, MAC, signatures, etc)
Client-side verifications
Setup and data upload
Authorization for TPA
Challenge for integrity proof
Proof verification
Updated data upload
Updated metadata upload
Verification of updated data
Verification framework
Simple data
Data with multiple replicas
Distributed data
Unstructured data
Cloud data storage for verification
Categorized data
Shared data
......

Fig 2: A brief overview of integrity verification over big data in cloud computing -- lifecycle and research topics

small but very frequent. For example, in 2010 Twitter is producing every day up to 12 terabytes of data, composed of tweets with a size of 140 characters maximum [17]. 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 are not only in security, but also in good efficiency.

III. Problem Analysis -- Framework and Lifecycle

There are 3 participating parties in an integrity verification scheme: 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. Although the three forms a robust and efficient triangle, each of the two parties are only semi-trusted by each other as shown in Fig.1. New security exploits may appear while verifying data integrity, which is why we need a good framework to address this problem systematically. 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 verification -> Updated data upload -> Updated metadata upload -> Verification of updated data

The relationship and order of these steps are illustrated in Fig. 2. 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 [13]. Then, these metadata will be uploaded and stored alongside with 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 plaintext extraction.

Challenge and verification of data storage: This step is where the main requirement -- integrity verification -- to be fulfilled. The client will send a challenge message to the server, and server will compute a response over the pre-stored data (HLA) and the challenge message. 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 insert, delete 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), according 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 have been performed successfully in order to ensure the updated data can still be verified correctly in the future.

We will show in the next section how each steps in this lifecycle was developed and evolved by analyzing some representative approaches in this research area.

IV. Representative Approaches and Analysis

We first introduce the basic idea behind the designs as well as some common notations. 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 (or more 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.

A. 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.

1) RSA Signature:

The RSA signature is classic and one of the earliest signature schemes under the scope 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.

2) Bilinear Pairing and BLS Signature:

BLS signature is proposed by Boneh, Lynn and Shacham [7] 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 $e: G \times G \to G_T$, where G_T is a multiplicative cyclic group with prime order1. A usable e should have the following properties: bilinearity – $\forall\, m, n \in G \Rightarrow e(m^a, n^b) = e(m, n)^{ab}$; 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 $e: G_1 \times G_2 \to G_T$ may also be applied, as was pointed out in [7].

Based on a bilinear map $e: G \times G \to G_T$, 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 $e(\sigma, g) = e(h(m), y)$.

3) Merkle Hash Tree:

The Merkle Hash Tree (MHT) [16] 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 a 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 $N_P = \{h(\mathcal{H}_1 || \mathcal{H}_2)\}$ 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 every of its upper level so that the root value R can be computed through $\{m_i, \Omega_i\}$. For example, for the MHT demonstrated in Fig.3, m_1's AAI $\Omega_1 = \{h(m_2), h(e), h(b)\}$.

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

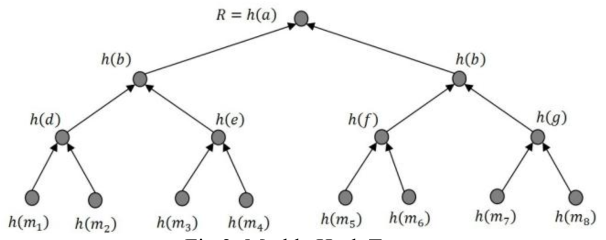

Fig 3: Merkle Hash Tree

B. Representative Schemes

Now we start to introduce and analyze some representative schemes. Note that all computations are within the cyclic group $\mathbb{Z}_p$ or $\mathbb{Z}_N$.

1) PDP

Proposed by Ateniese, et, al. in 2007, PDP (provable data possession) can provide authors with efficient verification over their outsourced data storage [3, 4]. 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 $\sigma_i = (h(v||i) \cdot g^{m_i})^d$. 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 $g_s = g^s \bmod N$ 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 $\rho = H(\tau^s \bmod N)$.

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. 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.

2) Compact POR

Compact POR is proposed by Shacham, et, al. in 2008 [20]. Compared to original POR, the authors provided an improved rigorous security proof.

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 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 $e: G \times G \rightarrow G_T$ 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 $e(\sigma, g) = e(\prod_i (H(i)^{v_i}) \cdot u^\mu, v)$.

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.

3) DPDP

DPDP (Dynamic PDP), proposed in 2009, is the first integrity verification scheme to support full data dynamics [11]. It is from here that the processes in integrity verification schemes started to form a lifecycle. They utilized another authenticated data structure -- rank-based 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) 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 [28]. They term the latter as 'public auditability', as the verification is often done by a sole-duty third-party auditor (TPA).

A 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 $e(\sigma, g) = e(\prod_i (H(m_i)^{v_i}) \cdot u^\mu, v)$, TPA will first verify $H(R)$'s signature to ensure 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 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.

There was also a follow-up work to improve this scheme for privacy preserving public auditing [24]. 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.

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. [15] proposed a public auditing scheme with 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 [25]) 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 $u_j \in U, U = \{u_k \in \mathbb{Z}_p\}, k \in [1, s_{max}]$ 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. 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 $e(\sigma, g) = e(\omega, \upsilon)$, 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 this 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 retrieved and transferred back to the client for tag re-computation. Experimental results have also demonstrated some significant efficiency improvements.

V. 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. [14], 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 [6] 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 [8]. This is also a practical solution, because cloud will 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 [5].

There is also some work proposed in the most recent years. Based on previous work and the recent developments of big data and cloud, they can be the more practical solution for specific cloud environments and applications. As mentioned before, [15] is a good example. For big enterprises, data migration is a big problem in the adoption process of cloud, because 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 [31] for verification of data stored in separated domains. As cloud data sharing becomes a hot topic, Wang et, al. worked on secure data verification of shared data storage [22] and also with efficient user management [23]. Zhang et, al. proposed a scheme with a new data structure called update tree [30]. Without conventional authenticated data structures such as MHT, the proposed scheme has a constant proof size and support fully data dynamics. However, the scheme does not support public verification/auditing at the moment.

VI. Conclusions and Future Directions

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, we are particularly interested in looking at the following aspects.

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 aspects, 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.

Security: Security is always a problem between spear and shield; that is, attack and defend. 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.

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 [31]. On the other hand, elasticity is one of a biggest reason why big companies are moving their business, especially service-related business, to the cloud [1]. 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 a cloud environment.

References

[1] Customer Presentations on Amazon Summit Australia, Sydney, 2012 Available: http://aws.amazon.com/apac/awssummit-au/, accessed on 25 August, 2013.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Communications of the ACM,* vol. 53, pp. 50-58, 2010.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote Data Checking Using Provable Data Possession," *ACM Transactions on Information and System Security,* vol. 14, p. Article 12, 2011.

[4] G. Ateniese, R. B. Johns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and 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

[5] G. Ateniese, S. Kamara, and 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.

[6] G. Ateniese, R. D. Pietro, L. V. Mancini, and 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.

[7] D. Boneh, H. Shacham, and B. Lynn, "Short Signatures from the Weil Pairing," *Journal of Cryptology,* vol. 17, pp. 297-319, 2004.

[8] R. Curtmola, O. Khan, R. C. Burns, and 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.

[9] 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 25 August, 2013

[10] 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 25 August, 2013

[11] C. Erway, A. Küpçü, C. Papamanthou, and 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.

[12] 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 25 August, 2013.

[13] R. Johnson, D. Molnar, D. Song, and D. Wagner, "Homomorphic Signature Schemes," *Topics in Cryptology - CT-RSA 2002, Lecture Notes in Computer Science,* vol. 2271, pp. 244-262, 2002.

[14] A. Juels and 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.

[15] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Ramamohanarao, "Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-grained Updates," *IEEE Transactions on Parallel and Distributed Systems,* 2013.

[16] 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.

[17] 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 25 August, 2013.

[18] S. Nepal, S. Chen, J. Yao, and 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.

[19] S. E. Schmidt. *Security and Privacy in the AWS Cloud*. Available: http://aws.amazon.com/apac/awssummit-au/, accessed on 25 August, 2013.

[20] H. Shacham and 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

[21] C. Vecchiola, R. N. Calheiros, D. Karunamoorthy, and R. Buyya, "Deadline-driven Provisioning of Resources for Scientific Applications in Hybrid Clouds with Aneka," *Future Generation Computer Systems,* vol. 28, pp. 58-65, 2012.

[22] B. Wang, S. S. M. Chow, M. Li, and 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.

[23] B. Wang, B. Li, and 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.

[24] C. Wang, S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Transactions on Computers,* 2011.

[25] C. Wang, S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Transactions on Computers,* vol. 62, pp. 362-375, 2013.

[26] L. Wang, M. Kunze, J. Tao, and G. v. Laszewski, "Towards Building A Cloud for Scientific Applications," *Advances in Engineering Software,* vol. 42, pp. 714-722, 2011.

[27] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and 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.

[28] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 22, pp. 847 - 859, 2011.

[29] J. Yao, S. Chen, S. Nepal, D. Levy, and 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.

[30] Y. Zhang and M. Blanton, "Efficient Dynamic Provable Possession of Remote Data via Update Trees," 2012.

[31] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage," *IEEE Transactions on Parallel and Distributed Systems,* vol. 23, pp. 2231-2244, 2012.

[32] D. Zissis and D. Lekkas, "Addressing Cloud Computing Security Issues," *Future Generation Computer Systems,* vol. 28, pp. 583-592, 2011.