Enabling Fast Public Auditing and Data Dynamics in Cloud Services

Changhee Hahn, Hyunsoo Kwon, Daeyeong Kim, and Junbeom Hur

Abstract—Public auditing enables efficient integrity checks of data assigned to cloud servers. In this paper, we revisit the public auditing for encrypted data, in which a major concern is how to effectively support data dynamics, i.e., data modification, insertion, and deletion. We first determine which factor in existing auditing schemes most limits data dynamics from a cost perspective. We then propose a novel public auditing scheme that provides data dynamics that are orders of magnitude faster than previous methods. Our novel auditing challenge-response protocol reduces the computation cost of the TPA significantly, thus increasing the verification speed for the auditing results. Performance and security analysis demonstrates that the proposed scheme generates minimal computation costs while guaranteeing data integrity and privacy against an untrusted cloud.

Index Terms—Public auditing, data dynamics, cloud computing.

1 Introduction

Hosting data in the cloud minimizes maintenance requirements, allowing users to easily access their data on cloud servers [1]. However, cloud servers have full control over outsourced data, which raises security concerns about data integrity [2], [3]. The cloud, for example, might have the financial incentive to discard rarely accessed data, freeing up valuable storage space to, say, host other data-centric applications [4]. Therefore, users need to confirm periodically that their data is intact but this has become increasingly onerous due to the ever-growing volume of data being outsourced.

Public auditing addresses this issue by utilizing a third-party auditor (TPA) to verify the correctness of the data in the cloud on behalf of users [5], [6]. Public auditing works by dividing a file into many blocks, with each block associated with auditing metadata. It is especially useful in terms of integrity checks because testing only a few blocks is sufficient to verify file integrity.

As more cloud-backed services adopt dynamic file updates [25], public auditing is required to support data dynamics such as block-wise modification, insertion, and deletion. Unfortunately, we have observed that previous schemes support data dynamics at a non-trivial cost [12], [13], [40]. Specifically, performing an operation on a single block requires a significant volume of auditing metadata associated with other blocks to be updated, delaying integrity checks for dynamic file updates. For example, suppose a data owner uses online collaboration services to share a source code with a group of users. As shown in Figure 1, the data owner splits the code into a fixed number of lines and group them as a set, with each set associated with a tag as auditing metadata. In this case, inserting even a single line to a specific position of a set can affect the subsequent sets, which incurs not only the re-computation of tag associated with the set, but also re-computation of all tags corresponding to the subsequent sets. As the shared codes can be frequently updated, efficient data dynamics for outsourced data must be taken into account for flexible service provisioning.

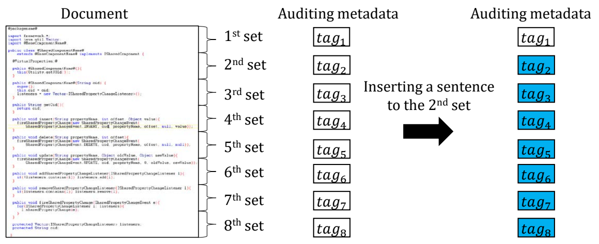


Fig. 1: Illustration of data dynamics for collaborative code editing. Tags in the blue boxes are newly computed after inserting a new line to the second set.

It is also desirable for a TPA to rapidly complete the verification and notify the user of the results. However, according to our measurements, the TPA experiences a delay of approximately 1.0 to 1.6 seconds when verifying the integrity of a file [12], [13], [40]. This lag may not be acceptable because auditing requests can be concentrated within a specific time period. Typically a TPA is assigned numerous files, so it would be highly advantageous if the TPA-side computation cost with respect to verification can be reduced.

In this paper, we revisit public auditing with data dynamics to investigate the factor in exiting auditing schemes that most restricts data dynamics in terms of cost. We find that auditing metadata m_i of block b_i in an n-block file is tightly coupled with its index i. We observed that this coupling is necessary to prevent cloud misbehavior. Specifically, auditing a file involves (b_i, m_i) pairs in response to a random selection of indices i as an auditing request. The use of pairs that do not correspond to these indices leads to verification failure. Unfortunately, this coupling is inefficient in terms of data dynamics because an operation,

- *C. Hahn, H. Kwon, D. Kim, and J. Hur are with the Department of Computer Science and Engineering, Korea University, Seoul, Korea. E-mail: jbhur@korea.ac.kr*

TABLE 1: Summary of data dynamics for public auditing

Scheme		Wang et al. [12]	Liu et al. [13]	Guo et al. [40]	Proposed scheme
Dynamic operations	Modification	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
	Insertion	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
	Deletion	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$

say an insertion, at the i-th position requires the replacement of the previous auditing metadata $(m_i, m_{i+1}, ..., m_n)$ with new metadata $(m'_{i+1}, m'_{i+2}, ..., m'_{n+1})$.

To address this issue, we carefully decouple the auditing metadata and index using a novel auditing side information approach. Specifically, for each block b_i, we generate auditing metadata and auditing side information (*SI*), both of which are detached from the index but cryptographically coupled to each other. The *SI* helps the TPA to detect cloud misbehavior, such as submitting pairs that do not correspond to the indices. At the same time, extremely rapid data dynamics is possible because the auditing metadata is no longer associated with the index. Our experiment demonstrates that data dynamics required approximately 7 milliseconds in the proposed scheme, which is orders of magnitude faster than 3 to 13 seconds required for previous schemes [12], [13], [40]. We summarize the asymptotic costs for data dynamics for an n-block file in Table 1.

Our novel auditing challenge-response protocol reduces the computation cost to the TPA significantly. Specifically, the TPA-side computation cost with respect to verification is a constant number of pairings and exponentiations in a cyclic group, while prior works require those operations linearly with the number of challenged blocks. The proposed scheme facilitates pre-computation capabilities such that, after sending an auditing request to the cloud, the TPA can pre-compute all exponentiation operations needed for the subsequent phase. Note that these computations can be performed in parallel with the cloud's efforts to compute the auditing response. According to our experiment, the TPA can notify users of the auditing results within 4 milliseconds, while previous schemes require 1.0 to 1.6 seconds [12], [13], [40].

Lastly, the proposed scheme is compatible with any symmetric-key encryption algorithm such that the blocks are encrypted any encryption algorithm of the data owner's choice. Data confidentiality is preserved against the cloud due to the CPA-secure property of the underlying encryption algorithm. We prove that the cloud cannot learn the outsourced data during the auditing process, and the cloud cannot forge a valid proof in response to the auditing request from the TPA.

Contribution. Overall, our contributions in this paper are as follows.

- We propose a novel public auditing scheme for encrypted data that supports extremely fast data dynamics. Asymptotic analysis demonstrates that the proposed scheme has $\mathcal{O}(1)$ complexity, while that of previous schemes ranges from $\mathcal{O}(\log n)$ to $\mathcal{O}(n)$ [12], [13], [40].
- Our novel auditing challenge-response protocol reduces the computation cost of the TPA significantly, thus increasing the verification speed for the auditing results.
- We analytically detail the performance of our constructions over prior work and experimentally confirm that the proposed data dynamics method is orders of magnitude faster than that of previous schemes.
- We formally define the security model under which we rigorously prove security of the proposed scheme to show that data integrity and privacy is preserved in the presence of an untrusted cloud.

The rest of this paper is as follows. We begin with a discussion of related work in Section 2. In Section 3, we provide a system description and outline the design goals. Section 4 presents the construction of the proposed scheme in detail, followed by security and performance analyses in Section 5. We conclude the paper in Section 6.

2 Related Work

In this section, we briefly review previous auditing studies, focusing on public auditability, data confidentiality, and data dynamics for outsourced data. In line with the purpose of data outsourcing, public auditability is a desirable property of pragmatic cloud environments, in which data owners can be relieved from the burdensome task of data management. The confidentiality of outsourced data is also an important consideration given previous data breaches in cloud storage [2], [3]. Unlike conventional online storage services, outsourced data in cloud storage can be frequently updated, as with Twitter [21]. Therefore, data dynamics for outsourced data have to be taken into account for flexible service provisioning. Unfortunately, previous techniques do not satisfy the all three of these properties.

2.1 Limited Operations for Outsourced Data

Ateniese et al. [5] proposed Provable Data Possession (PDP), in which a public verifier can check the correctness of a user's stored data in the cloud. PDP utilizes an RSA-based Homomorphic Linear Authenticator (HLA) for outsourced data. Because an HLA can be aggregated, it is possible to compute an aggregated HLA that authenticates a linear combination of individual data blocks. Using sampling strategies, the public verifier is able to audit the integrity of the outsourced data without retrieving the entire data set. However, this scheme does not consider dynamic operations for outsourced data. To support dynamic operations, Ateniese et al. [6] designed an improved PDP scheme using symmetric keys and a cryptographic hash function. However, this scheme only supports a limited number of verification challenge queries. In addition, it does not support block insertion, though append-type insertion is possible.

Juels and Kaliski [7] defined another scheme called Proofs of Retrievability (POR). The POR scheme incorporates special blocks called sentinels, which are randomly embedded into the data for detection purposes. However, they restrict operations on the updated data. Although this scheme ensures retrievability of a file using error-correcting codes and spot-checking, it has a drawback in that the number of challenge queries is fixed. To address these issues, Shacham and Waters [8] proposed an improved POR scheme based on a Boneh-Lynn-Shacham (BLS) signature [24]. It overcomes the limitation of the POR scheme in terms of the number of challenge queries and provides proof of security. However, it only considers static data files because the cloud cannot distinguish the relationship between the data blocks and encrypted codewords. Liu et al. [16] proposed a scheme based on regenerating code, which enables a user to verify the integrity of random subsets of outsourced data against corruption. However, this scheme does not support dynamic data operations.

2.2 Supporting Data Dynamics

Various auditing schemes have been proposed to support data dynamics. However, to the best of our knowledge, none of these have been able to achieve data dynamics that are as rapid as in our proposed scheme. In this subsection, we discuss public auditing with data dynamics which selectively preserves privacy against the TPA and/or the cloud. We believe that there are no technical difficulties associated with achieving data privacy in public auditing. It can be attained by encrypting each block and generating the auditing metadata corresponding to the ciphertext. Nevertheless, we classify the existing auditing schemes in terms of privacy for clarity. We start by describing public auditing that does not consider privacy.

2.2.1 Auditing Without Privacy

Erway et al. [9] designed the Dynamic Provable Data Possession (DPDP) scheme, a dynamic version of PDP, by supporting the updating of stored data. It uses a rank-based authenticated skip list to authenticate the tag information of challenged or updated data blocks before the verification procedure. Wang et al. [10] proposed a public auditing scheme that combines an HLA with a Merkle Hash Tree (MHT) to support dynamic data operations. However, in this scheme, the MHT needs to be re-constructed once the data has been updated. Zhu et al. [11] proposed a dynamic auditing scheme for cloud data based on a fragment structure, random sampling, and index hash tables. Their scheme is similar to ours in the sense that it does not involve the coupling of the auditing metadata and the index. They view a file as a group of sectors, where a set of sectors form a block on which auditing metadata is generated. Unfortunately, if any data dynamics for a sector occurs, every subsequent sector is affected. This can lead to the replacement of previous auditing metadata with new metadata, thereby resulting in inefficient data dynamics.

Another line of researches that aimed at reducing the cost relating to data dynamics are batch update [40], index switcher [19], and dynamic hash table [14], [20]. Specifically, a batch update of data dynamics is to reduce the amount of computation cost relating to dynamic data updates by performing and verifying multiple update operations at once [40]. Although it may avoid repetitive computations relating to data dynamics, the update delay can be unacceptable in time-critical applications. The index switcher approach provides a way of switching the encoded indices between block indices and tag indices [19], while dynamic hash table based approaches aim to decouple the indices from tags [14], [20]. However, all of these schemes incur a significant trade-off: the TPA-side computation cost relating to verification increases linearly with the number of challenged blocks. Given c number of challenged blocks, the TPA computes c pairings [14] or c exponentiations in a cyclic group [19], [20]. By contrast, the proposed scheme only requires two pairings and two exponentiations regardless of the number of challenged blocks.

2.2.2 Auditing with Privacy

Juels et al. showed how POR can provide public auditing while preserving data privacy against the TPA [7]. Specifically, a POR client can separate data encryption and verification privileges, and delegate the verification capability to the TPA for public verification. Wang et al. [12] proposed a privacy-preserving public auditing scheme using random masking to prevent data content from being disclosed to the TPA. Similarly, Liu et al. [13] proposed a secure and efficient public auditing scheme using a homomorphic hash function and random masking. Shah et al.'s auditing scheme uses standard encryption (e.g. AES), but their scheme requires decryption in the cloud before any further computations can be performed [15].

Ramaiah et al. [17] proposed a privacy-preserving public auditing scheme, which encrypts each data block using somewhat homomorphic encryption. Thus, both the cloud and the TPA are not able to learn the content of a user's data while enabling integrity check of cloud data. However, this scheme only offers limited data dynamics, such as block modification and appending.

2.3 Resilience Against Malicious Entities

While many prior works relied on the trust assumption that the TPA and the clients are honest, several subsequent works attempted to relax such an assumption for allowing the TPA and the clients to deviate from the protocol. Armknecht et al. proposed outsourced POR (OPOR) [27], in which a POR client outsources the auditing workload to the untrusted TPA and verifies the TPA's work. As the security of the POR challenge-response protocol depends on the properly sampled challenges (e.g., a set of random values), a challenging problem is how to ensure the correct challenge sampling (and sharing) between the untrusted TPA and the client. OPOR addressed this issue by means of blockchain networks: both the TPA and the client use the public blockchain to extract (synchronized) seed values for pseudo-randomness to sample the challenges. Zhang et al. [28] also utilized the blockchain with a more relaxed trust assumption, showing how to audit data in the presence of multiple malicious clients.

2.4 Recent Studies and Limitations

Recent proposals aim to support additional functionalities on top of public auditing. Wang et al. [18] proposed a verifiable encrypted outsourced database scheme based on Bloom filters. This scheme allows the user to ensure the correctness of a search result by checking whether the search request is part of the Bloom filters. However, this scheme does not consider dynamic data operations. Some studies on public auditing have focused on resilience to key leakage [32], public auditing of shared data [35], and blockchain-based auditing [34], [36], [37]. Although these schemes enhance either security and/or scalability in terms of multiple users, efficient data dynamics has not yet been achieved.

Erway et al. improved a dynamic PDP scheme in which data dynamics is supported, but its communication and computation complexities are $\mathcal{O}(\log n)$, where n denotes the number of blocks in a file [39]. Moreover, the data owner should engage in verifying the integrity of the outsourced data. Guo et al. addressed this by enabling the TPA to audit the data on behalf of the data owner, in which the cost of data dynamics remains $\mathcal{O}(\log n)$ [40].

As a hardware-aided solution, auditing metadata could be securely computed on untrusted cloud servers using trusted execution environments like Intel Software Guard Extensions (SGX) [30]. In this approach, the computation of auditing metadata can be securely performed with a single machine in the cloud as long as the cloud server is SGX-enabled. However, the workload remains the same in terms of computation. Users would be reluctant to use *computationally intensive* auditing because cloud computing vendors typically charge for resource usage with pay-as-you-go pricing [31]. Therefore, an important step toward the more widespread use of auditing is to establish cost-effective data dynamics.

3 System Description and Goals

In this section, we describe the definition of public auditing for encrypted data, followed by the goals of the proposed scheme.

3.1 Public Auditing for Encrypted Data

Public auditing for encrypted data provides efficient integrity checking of the encrypted dataset outsourced to a remote server. The system model of public auditing for encrypted data involves three entities as follows:

- The user, who has a large volume of data to be stored in the cloud and hopes to maintain data privacy and integrity against the cloud.
- The cloud, which provides data storage services and computing resources for the user's data.
- The TPA, which checks the integrity of the data stored in the cloud via a challenge-and-response protocol with the cloud on behalf of the user.

By storing the data in the cloud, the user can be relieved of the burden of storage and computation. He can also access and update his stored data for various purposes. To reduce computation costs and eliminate the requirement of the user always being online, the user can employ the TPA while hoping to protect the privacy and integrity of his outsourced data.

The syntax of public auditing for encrypted data is as follows:

Definition 1. *Public auditing for encrypted data consists of six algorithms:*

- $pp \leftarrow$***Setup***(λ): *It takes as input security parameter* λ *and returns public parameter* pp.
- $(sk, pk) \leftarrow$***KeyGen***(pp): *It takes as input* pp *and returns secret and public key pair* (sk, pk).
- $(\sigma, SI) \leftarrow$***SigGen***$(pk, sk, Enc(F))$: *It takes as input* pk, sk, *and ciphertext* $Enc(F)$ *of file* F, *where* $Enc(\cdot)$ *is a CPA-secure symmetric encryption algorithm (e.g., AES-GCM). It returns tag* σ *and auditing side information* SI.
- $chal \leftarrow$***Challenge***(pk, id, SI): *It takes as input* pk, *identifier* id *of a file to audit, and* SI. *It returns challenge message* $chal$.
- $proof \leftarrow$***ProofGen***$(pk, Enc(F), \sigma, chal)$: *It takes as input* $pk, Enc(F), \sigma$, *and* $chal$. *It returns response message* $proof$.
- $\{True, False\} \leftarrow$***ProofVrfy***$(pk, proof, id, SI)$: *It takes as input* $pk, proof, id$, *and* SI. *It returns verification result* $True$ *if auditing succeeds; it returns* $False$ *otherwise.*

Correctness. The correctness of public auditing for encrypted data guarantees that the TPA always accepts the proof submitted by an honest server. More formally, for all $pp \leftarrow$ **Setup**(λ), $(sk, pk) \leftarrow$ **KeyGen**(pp), $(\sigma, SI) \leftarrow$ **SigGen**$(pk, sk, Enc(F))$, $chal \leftarrow$ **Challenge**(pk, id, SI), and $proof \leftarrow$ **ProofGen**$(pk, Enc(F), \sigma, chal)$, it holds with overwhelming probability that $True \leftarrow$ **ProofVrfy**$(pk, proof, id, SI)$.

Soundness. The soundness of public auditing for encrypted data guarantees that the server can convince the TPA if and only if it stores the (encrypted) file intact. Stated differently, the server cannot generate a valid proof without storing the file intact. We discuss how to formally capture the soundness property and prove it under a well-known complexity problem in Section 4.2.4.

Threat Model. We consider two threat models with regard to data integrity and privacy.

– Data integrity: In the cloud storage, a malicious hosting service provider may discard data that has not been accessed or that is rarely accessed to save storage space. In addition, the hosting service provider may tamper with sensitive user data affected by Byzantine failures and management errors in order to maintain its reputation.

– Data privacy: The cloud server may attempt to leak the content of the stored data. It may also try to recover the data during the auditing process.

The TPA is not necessarily trusted in terms of data privacy but trustworthy in terms of data integrity. Considering the threat models, we pursue the following security goals. First, if a malicious cloud provider forges or discards user data, it will not be able to produce a valid proof (i.e., the soundness is preserved). Second, the cloud cannot uncover the user's data content during the auditing process.

3.2 Design Goals

The proposed public auditing scheme for encrypted data should have the following properties:

– Public auditing: The TPA can periodically check the integrity of the data on behalf of the user without retrieving a copy of the entire data set.

– Storage correctness: The cloud can pass TPA verification if and only if it keeps the data intact.
– Privacy preservation: During the auditing process, the cloud server cannot obtain the user's plain data.
– Support for data dynamics: The user is able to perform block-level operations on the outsourced data, including block modification, insertion, and deletion.

4 Proposed Scheme

In this section, we present a public auditing scheme designed to meet the aforementioned design goals. After introducing the preliminaries, we provide an overview of the proposed scheme. We then describe the construction of our main scheme and outline how it works in supporting dynamic operations.

4.1 Preliminaries

4.1.1 Bilinear Maps

Let **GroupGen** be a probabilistic polynomial time (PPT) algorithm that uses the security parameter 1^λ as input, and outputs a description $(p, \mathbb{G}_1, \mathbb{G}_2, g, e)$ of pairing groups. $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order p, while g is the generator of $\mathbb{G}_1$. The bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ must satisfy the following properties:

- Bilinearity: $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in \mathbb{G}_1$ and a, $b \in \mathbb{Z}_p$.
- Non-degeneracy: There are $u, v \in \mathbb{G}_1$ such that $e(u, v) \neq 1$.
- Computability: There is an efficient algorithm to compute $e(u, v)$ for any $u, v \in \mathbb{G}_1$.

4.1.2 Symmetric Encryption

Symmetric encryption **SE** uses the same secret key for encryption and decryption, and guarantees confidentiality for messages. In this paper, **SE** can be instantiated by any CPA-secure symmetric encryption algorithm such as AES-GCM. **SE** consists of the following algorithms:

- $k \leftarrow$ **SE.gen**$(\mathcal{K})$: This algorithm takes as input a key space $\mathcal{K}$ from which it samples a secret key k uniformly at random.
- $ct \leftarrow$ **SE.enc**(k, m): This algorithm takes as input k and a message m, and returns a ciphertext ct.
- $m \leftarrow$ **SE.dec**(k, ct): This algorithm takes as input k and ct, and returns m.

In what follows, we omit the secret key k in describing **SE.enc**(k, m) such that **SE.enc**(m) for brevity.

4.1.3 Homomorphic Hash Function

A homomorphic hash function [22], [23] is a hash function that satisfies two properties:

- Homomorphism: For any two messages m_1, m_2 and scalars α_1, α_2, it holds that
$$H(\alpha_1 m_1 + \alpha_2 m_2) = H(m_1)^{\alpha_1} \cdot H(m_2)^{\alpha_2}.$$
- Collision Resistance: There is no probabilistic polynomial-time (PPT) adversary capable of forging $(m_1, m_2, m_3, \alpha_1, \alpha_2)$ to satisfy both
$$\begin{aligned} m_3 &\neq \alpha_1 m_1 + \alpha_2 m_2, \\ H(m_3) &= H(m_1)^{\alpha_1} \cdot H(m_2)^{\alpha_2}. \end{aligned}$$

4.2 Scheme Construction

4.2.1 Overview

As shown in Figure 2, the proposed scheme consists of the three phases:

- **Data upload**: This phase is run mostly by the user. The user generates public and secret parameters. He divides a file into multiple data blocks before storing it in the cloud. To ensure data confidentiality, the user needs to encrypt the data blocks. To enable the TPA to audit without exposing the key to it, the user computes auditing metadata that feature homomorphic hash properties. The user sends encrypted data blocks and the corresponding auditing metadata to the cloud. He then deletes the data blocks and the auditing metadata from the local storage.
- **Data update**: In this phase, we assume that the user downloaded some files of interest in advance. If he finds that some blocks of a file need to be updated (e.g., block modification, insertion, and deletion), he encrypts the updated block and generates new auditing metadata corresponding to the updated block[1]. Then, he uploads this to the cloud.
- **Data auditing**: This phase is run interactively between the user, the TPA, and the cloud. When the user wants to check the integrity of the data stored in the cloud, the user sends an auditing request to the TPA. Upon receiving the auditing request from the user, the TPA generates and sends a challenge message to the cloud server. The cloud server derives *proof* from the stored data and sends it back to the TPA. Finally, upon receiving *proof* from the cloud, the TPA verifies the correctness of the *proof*. If the verification succeeds, it indicates that the integrity of the file is intact; otherwise, the file is compromised. We note that this phase can run without the user: the TPA periodically audit data of its choice by interacting with the cloud.

4.2.2 Our Scheme

The proposed scheme consists of the following algorithms: **Setup**, **KeyGen**, **SigGen**, **Challenge**, **ProofGen**, and **ProofVrfy**.

$pp \leftarrow$**Setup**(λ): The user runs the setup algorithm, which takes as input the security parameter λ. He runs **GroupGen** to obtain $(p, \mathbb{G}_1, \mathbb{G}_2, g, e)$. He also defines the descriptions of a homomorphic hash function $H : \mathbb{Z}_p \rightarrow \mathbb{G}_1$ and a hash function $\mathcal{F} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$. Lastly, he sets and returns a public parameter $pp = (p, \mathbb{G}_1, \mathbb{G}_2, g, e, H, \mathcal{F})$.

$(sk, pk) \leftarrow$**KeyGen**(pp): The user runs the key generation algorithm, which takes as input pp. He chooses $x \in_R \mathbb{Z}_p^*$ and computes $v = g^x$. He sets the public and secret key as $pk = (pp, v)$ and $sk = x$, respectively.

$(\sigma, SI) \leftarrow$**SigGen**$(pk, sk,$ **SE.enc**$(F))$: The user runs the signature generation algorithm, which takes as input pk, sk, and **SE.enc**(F). Assume that the user encrypted the file F using a symmetric secret key $k \leftarrow$ **SE.gen**$(\mathcal{K})$

1. If the update corresponds to deletion, then the user deletes the block and does nothing more.

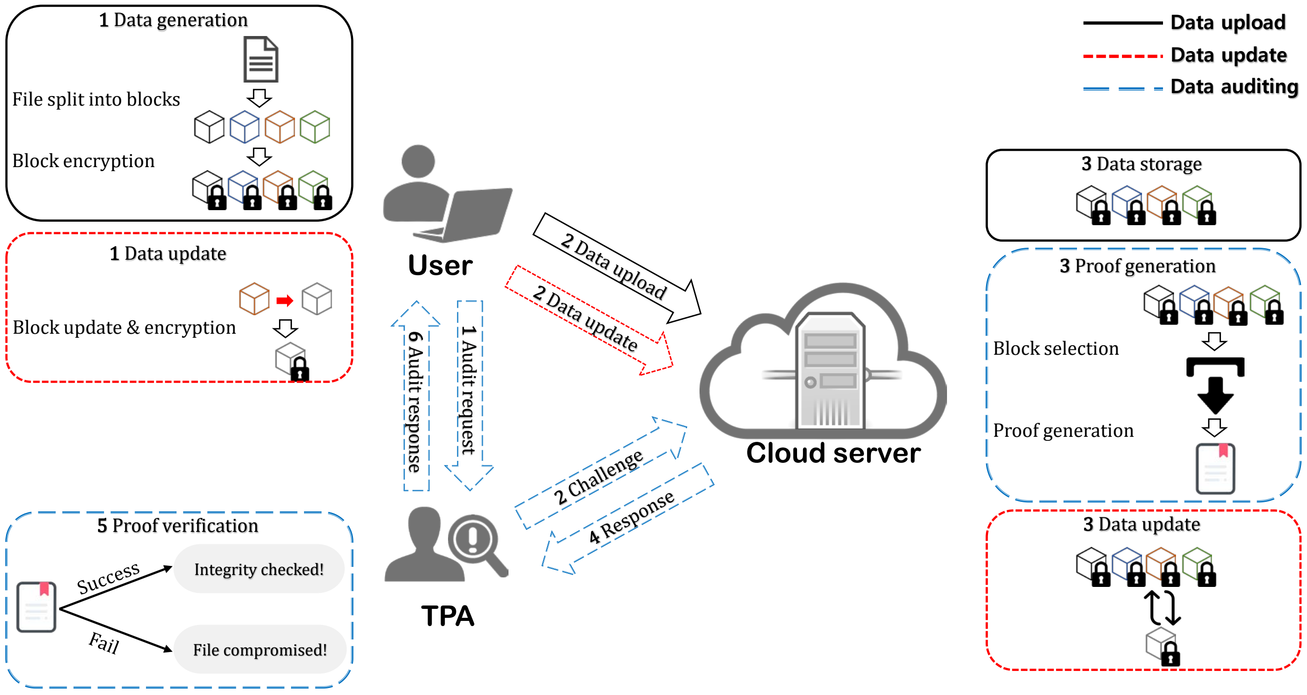


Fig. 2: Framework of the proposed scheme

in advance. Assume also that F was divided into n blocks as $F = \{m_1, \ldots, m_n\}$ such that $\mathbf{SE.enc}(F) = (\mathbf{SE.enc}(m_1), ..., \mathbf{SE.enc}(m_n))$. Let $ct_i = \mathcal{F}(\mathbf{SE.enc}(m_i))$ for $i \in \{1, ..., n\}$, and $id \in \mathbb{Z}_p^*$ be the identity of file F. He chooses two random coefficients $(\alpha, \beta) \in_R \mathbb{Z}_p$ and a random variable $s \in_R \mathbb{Z}_p$. He defines a set of seeds $\{s_i\}_{1 \le i \le n}$ from $\mathbb{Z}_p$ such that

$$s_1 = \alpha\beta s, s_2 = \alpha^2\beta^2 s, ..., s_n = \alpha^n\beta^n s.$$

Then, he computes the auditing side information $SI = \{\alpha, \beta, s, H(s_1), ..., H(s_n)\}$. For each $i \in \{1, \ldots, n\}$, the user computes the tag as

$$\sigma_i \leftarrow (H(id) \cdot H(ct_i + s_i))^x \in \mathbb{G}_1.$$

He then sets $\sigma = (\sigma_1, \ldots, \sigma_n)$ and sends $\{\mathbf{SE.enc}(F), \sigma\}$ to the cloud, and SI to the TPA. Finally, the user deletes the plain data file F from local data storage.

$chal \leftarrow$**Challenge**(pk, id, SI): The TPA runs the challenge generation algorithm, which takes as input *auditing request* from the user, as well as pk, id, and SI. To verify the integrity of the outsourced file of identity id, the user sends an *auditing request* to the TPA. Subsequently, the TPA chooses a random subset $I = \{idx_1, \ldots, idx_c\}$ from a set $\{1, ..., n\}$, where $c \ll n$. For each $idx_i \in I$, the TPA also chooses a random value $\nu_i \in \mathbb{Z}_p^*$, and generates $chal = \{(idx_i, \nu_i)\}_{idx_i \in I}$ as a challenge message. Note that $chal$ specifies the positions of the blocks to be checked. The TPA then sends $chal$ to the cloud. After sending the challenge message to the cloud, the TPA may compute $\prod_{i=idx_1}^{idx_c} H(id)^{\nu_i}$ and $\prod_{i=idx_1}^{idx_c} H(s_i)^{\nu_i}$ in advance. These values will be used in the **ProofVrfy** phase. Note that the computations of $\prod_{i=idx_1}^{idx_c} H(id)^{\nu_i}$ and $\prod_{i=idx_1}^{idx_c} H(s_i)^{\nu_i}$ only require two exponentiations in $\mathbb{G}_1$, respectively[2]. We explain why in Section 4.2.5.

$proof \leftarrow$**ProofGen**$(pk, \mathbf{SE.enc}(F), \sigma, chal)$: The cloud server runs the proof generation algorithm, which takes as input $pk, \mathbf{SE.enc}(F), \sigma$, and $chal$. The server parses $\mathbf{SE.enc}(F)$ as $(\mathbf{SE.enc}(m_1), ..., \mathbf{SE.enc}(m_n))$. For $i \in \{1, ..., n\}$, it computes $ct_i = \mathcal{F}(\mathbf{SE.enc}(m_i))$. Then, it computes

$$\mu = \sum_{i=idx_1}^{idx_c} \nu_i ct_i, \sigma = \prod_{i=idx_1}^{idx_c} \sigma_i^{\nu_i},$$

where μ and σ are the linear combination of the sampled blocks in the challenge message $chal$ and an aggregated tag, respectively. The cloud then sends $proof$=$\{\sigma, \mu\}$ to the TPA.

$\{True, False\} \leftarrow$**ProofVrfy**$(pk, proof, id, SI)$: The TPA runs the proof verification algorithm, which takes as input $pk, proof, id$, and SI. Upon receiving $proof$ from the cloud, the TPA computes $A = \prod_{i=idx_1}^{idx_c} H(id)^{\nu_i}$ and $B = \prod_{i=idx_1}^{idx_c} H(s_i)^{\nu_i}$, unless otherwise computed. Then, it checks the integrity of the outsourced data as follows:

$$e(\sigma, g) \overset{?}{=} e(A \cdot H(\mu) \cdot B, v).$$

The TPA returns $True$ if the above equation holds, and $False$ otherwise. The user is notified of the auditing result from the TPA.

2. Although multiplications in $\mathbb{G}_1$ and $\mathbb{Z}_p$ are required, the computational cost of them is negligible, compared with that of exponentiations in $\mathbb{G}_1$.

4.2.3 Correctness

Suppose for all $pp \leftarrow \mathbf{Setup}(\lambda)$, $(sk, pk) \leftarrow \mathbf{KeyGen}(pp)$, $(\sigma, SI) \leftarrow \mathbf{SigGen}(pk, sk, \mathbf{SE.enc}(F))$, and $chal \leftarrow \mathbf{Challenge}(pk, id, SI)$, the server honestly generates $proof \leftarrow \mathbf{ProofGen}(pk, \mathbf{SE.enc}(F), \sigma, chal)$. Then, the TPA obtains $True \leftarrow \mathbf{ProofVrfy}(pk, proof, id, SI)$ with overwhelming probability. More precisely,

$$\begin{aligned}
e(\sigma, g) &= e\left(\prod_{i=idx_1}^{idx_c} \sigma_i^{\nu_i}, g\right) \\
&= e\left(\prod_{i=idx_1}^{idx_c} \left(H(id) \cdot H(ct_i + s_i)\right)^{\nu_i}, g^x\right) \\
&= e\left(\prod_{i=idx_1}^{idx_c} H(id)^{\nu_i} \cdot \prod_{i=idx_1}^{idx_c} H\left(ct_i + s_i\right)^{\nu_i}, v\right) \\
&= e\left(A \cdot \prod_{i=idx_1}^{idx_c} H(\nu_i(ct_i + s_i)), v\right) \\
&= e\left(A \cdot H\left(\sum_{i=idx_1}^{idx_c} \nu_i(ct_i + s_i)\right), v\right) \\
&= e\left(A \cdot H\left(\sum_{i=idx_1}^{idx_c} \nu_i ct_i + \sum_{i=idx_1}^{idx_c} \nu_i s_i\right), v\right) \\
&= e\left(A \cdot H\left(\sum_{i=idx_1}^{idx_c} \nu_i ct_i\right) \cdot H\left(\sum_{i=idx_1}^{idx_c} \nu_i s_i\right), v\right) \\
&= e\left(A \cdot H(\mu) \cdot \prod_{i=idx_1}^{idx_c} H(\nu_i s_i), v\right) \\
&= e\left(A \cdot H(\mu) \cdot \prod_{i=idx_1}^{idx_c} H(s_i)^{\nu_i}, v\right) \\
&= e\left(A \cdot H(\mu) \cdot B, v\right).
\end{aligned}$$

4.2.4 Soundness

We formally capture the soundness property using the following auditing game:

Auditing Game:

- *Setup:* The challenger $\mathcal{C}$ runs $pp \leftarrow \mathbf{Setup}(\lambda)$ and $(sk, pk) \leftarrow \mathbf{KeyGen}(pp)$. $\mathcal{C}$ sends pk to the adversary $\mathcal{A}$ and keeps sk secret.
- *Query:* $\mathcal{A}$ makes **SigGen** queries adaptively: it chooses and splits a file F into n blocks of the same length, i.e., $F = \{m_1, ..., m_n\}$. It sends F to $\mathcal{C}$ who, upon receiving F, encrypts F using a CPA-secure symmetric encryption algorithm, which results in $Enc(F) = (Enc(m_1), ..., Enc(m_n))$. It runs $(\sigma, SI) \leftarrow \mathbf{SigGen}(pk, sk, Enc(F))$, where $\sigma = (\sigma_1, ..., \sigma_n)$ and $SI = (si_1, ..., si_n)$. Note that $Enc(m_i)$ corresponds to (σ_i, si_i), where $i \in \{1, ..., n\}$. $\mathcal{C}$ returns $(Enc(F), \sigma)$ to $\mathcal{A}$.

  $\mathcal{A}$ continues to query $\mathcal{C}$ for ciphertext and tag, which correspond to file F' of its choice and obtains $(Enc(F'), \sigma')$. $\mathcal{A}$ stores all the encrypted files and the corresponding tags in order.
- *Challenge:* $\mathcal{C}$ runs $\mathbf{Challenge}(pk, id, SI)$, where id is the identifier of file F. $\mathcal{C}$ obtains a challenge message $chal$, which contains a randomly chosen set of block indices $\{idx_1, ..., idx_c\}$ of F, where $c \ll n$ and $idx_i \in \{1, ..., n\}$ for $i \in \{1, ..., c\}$. $\mathcal{C}$ sends $chal$ to $\mathcal{A}$.
- *Forge:* $\mathcal{A}$ computes a response message $proof$ for the blocks of F, where each block corresponds to the block index specified by $chal$. $\mathcal{A}$ returns $proof$.
- *Audit:* If $\mathbf{ProofVrfy}(pk, proof, id, SI)$ returns $True$, then $\mathcal{A}$ wins the game; otherwise, $\mathcal{A}$ loses the game.

Definition 2. *A public auditing scheme for encrypted data is sound if the probability that any probabilistic polynomial time adversary wins the Auditing Game on a set of file blocks is negligible.*

To show that the server can convince the TPA if and only if it stores the (encrypted) file intact, we resort to the Decisional Diffie-Hellman (DDH) assumption [26], which posits as follows: Given the following set of elements $(g, g^a, g^b, z) \in \mathbb{G}_1^4$, where z is either g^{ab} or a random value R, algorithm $\mathcal{B}$, who returns 0 if $z = g^{ab}$ and 1 otherwise, has advantage ϵ in solving DDH in $\mathbb{G}_1$ if

$$\left|\Pr[\mathcal{B}(\mathbf{y}, z = g^{ab}) = 0] - \Pr\left[\mathcal{B}\left(\mathbf{y}, z = R\right) = 1\right]\right| \leq \epsilon,$$

where $\mathbf{y} = \left(g, g^a, g^b\right) \in \mathbb{G}_1^3$, and the probability is over the random choice of generator g in $\mathbb{G}_1$, the random choice of $a, b \in \mathbb{Z}_p$, and the random bits used by $\mathcal{B}$.

In Section 5.1, we prove the following theorem:

Theorem 1. *Under the DDH assumption, the proposed scheme guarantees soundness in the random oracle model.*

4.2.5 Pre-computable Verification

In the proposed scheme, the computationally extensive operations in the verification algorithm are the two pairings and two exponentiations in $\mathbb{G}_1$, where the computations of $\prod_{i=idx_1}^{idx_c} H(id)^{\nu_i}$ and $\prod_{i=idx_1}^{idx_c} H(s_i)^{\nu_i}$ correspond to the latter and are pre-computable. Specifically, these operations are independent of $proof$, so the TPA can initiate the verification phase beforehand. This reduces the verification time. We will discuss the actual cost in Section 5.2.

To see why computing $\prod_{i=idx_1}^{idx_c} H(id)^{\nu_i}$ and $\prod_{i=idx_1}^{idx_c} H(s_i)^{\nu_i}$ require only two exponentiations, we show how the TPA computes those values. First, to compute $\prod_{i=idx_1}^{idx_c} H(id)^{\nu_i}$, the TPA computes $\sum_{i=idx_1}^{idx_c} \nu_i$ and then computes $H(id)^{\sum_{i=idx_1}^{idx_c} \nu_i}$, which satisfies the following equation:

$$\begin{aligned}
H(id)^{\sum_{i=idx_1}^{idx_c} \nu_i} &= H(id)^{\nu_{idx_1}} \cdot H(id)^{\nu_{idx_2}} \cdots H(id)^{\nu_{idx_c}} \\
&= \prod_{i=idx_1}^{idx_c} H(id)^{\nu_i}.
\end{aligned}$$

Next, to compute $\prod_{i=idx_1}^{idx_c} H(s_i)^{\nu_i}$, the TPA computes $\sum_{i=idx_1}^{idx_c} (\nu_i \alpha^{(i)} \beta^{(i)})$. Then, the TPA computes $H(s)^{\sum_{i=idx_1}^{idx_c} (\nu_i \alpha^{(i)} \beta^{(i)})}$, where $\alpha^{(i)}$ and $\beta^{(i)}$ refer to the coefficients of s in s_i, respectively. Let L be $H(s)^{\sum_{i=idx_1}^{idx_c}(\nu_i \cdot \alpha^{(i)} \cdot \beta^{(i)})}$. Then, we have

$$\begin{aligned} L &= H(s)^{\nu_{idx_1}\alpha^{(idx_1)}\beta^{(idx_1)}} \cdots H(s)^{\nu_{idx_c}\alpha^{(idx_c)}\beta^{(idx_c)}} \\ &= H(\alpha^{(idx_1)}\beta^{(idx_1)}s)^{\nu_{idx_1}} \cdots H(\alpha^{(idx_c)}\beta^{(idx_c)}s)^{\nu_{idx_c}} \\ &= H(s_{idx_1})^{\nu_{idx_1}} \cdots H(s_{idx_c})^{\nu_{idx_c}} \\ &= \prod_{i=idx_1}^{idx_c} H(s_i)^{\nu_i}. \end{aligned}$$

4.3 Data Dynamics

The user may want to update his data dynamically when using the cloud storage system. We demonstrate how the proposed scheme supports data dynamics including block-level modification, insertion, and deletion. Figure 3 illustrates the process of dynamic updates in the proposed scheme and previous schemes such as [12], [13], [17], [40]. The computational complexity in terms of the modification of a block is similar for the proposed and previous schemes. However, when we insert or delete a block, the complexity of the previous schemes are $\mathcal{O}(n)$ [13], [17] and $\mathcal{O}(\log n)$ [12], [40], while that of the proposed scheme is $\mathcal{O}(1)$. This is because the auditing metadata of the previous schemes are tightly coupled with the indices. Thus, whenever the position of a block is moved, say from the i-th to the j-th index, the current metadata, which corresponds to i, should be generated again to correspond to j. However, the proposed scheme addresses this issue by decoupling indices from auditing metadata, while still enabling block-wise auditing via the auditing side information, which the TPA exploits to correctly audit data even when the blocks are updated.

4.3.1 Block Modification

When the user wants to modify a specific block of index i in the cloud, he first downloads the encrypted data block **SE.enc**(m_i) and corresponding tag σ_i from the cloud. The user then computes a new seed value $s'_i = \alpha \cdot s_i$ for the pseudorandom generator. He decrypts **SE.enc**(m_i) and modifies block m_i to m'_i, encrypts m'_i such that **SE.enc**(m'_i), and generates a tag σ'_i that corresponds to **SE.enc**(m'_i). Lastly, the user sends the index i, the new tag σ'_i, and the modified data block **SE.enc**(m'_i) to the cloud, and the auditing side information $H(s'_i)$ to the TPA. Upon receiving these, the cloud and the TPA replace the previous ones with the newly received ones.

4.3.2 Block Insertion

When the user inserts new block m between the j-th and $j+1$-th blocks, he sets a new seed $s'_{j+1} = \beta \cdot s_j$ and encrypts m such that **SE.enc**(m). He then computes a tag σ and auditing side information $H(s'_{j+1})$. The index information $j+1$, σ, and **SE.enc**(m) are sent to the cloud, and $(j+1, H(s'_{j+1}))$ to the TPA. Upon receiving these, the cloud inserts them into the $(j+1)$-th position. All subsequent blocks are moved accordingly. After the cloud updates the index table, the TPA confirms the update.

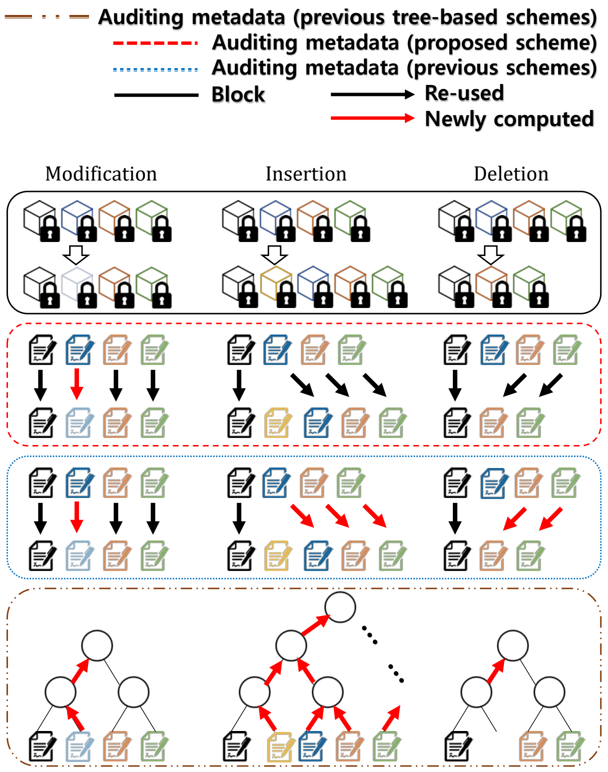

Auditing metadata (previous tree-based schemes)
Auditing metadata (proposed scheme)
Auditing metadata (previous schemes)
Block
Re-used
Newly computed
Modification
Insertion
Deletion

Fig. 3: Block-level dynamic updates

4.3.3 Block Deletion

Block deletion is the opposite of block insertion. When the user wants to delete the i-th block of a file in the cloud, he requests an update. The TPA updates its auditing side information by removing $H(s_i)$. The cloud deletes the corresponding block and the tag. All subsequent blocks are moved accordingly. After the cloud updates the index table, the TPA confirms the update.

5 Security And Performance Analysis

5.1 Security Analysis

In this section, we analyze the security of the proposed scheme with regard to data privacy and integrity.

In terms of data privacy, any attempt to leak data in the proposed scheme is reduced to breaking the underlying encryption algorithm **SE** as we use **SE** to encrypt the data. Since **SE** is CPA-secure and the user (who holds the symmetric secret key) never decrypts the ciphertexts outside his local storage throughout the protocol, it is immediate that the data privacy is preserved against the cloud server.

In terms of data integrity, we prove Theorem 1 as follows.

Proof. Throughout the proof, we reduce the security of the proposed scheme to the security of the DDH problem. Specifically, we assume that adversary $\mathcal{A}$ wins the Auditing Game with non-negligible probability. Then, we show how another adversary $\mathcal{B}$ uses $\mathcal{A}$ to solve the DDH problem without relying on the bilinear map. This approach is similar to the trapdoor DDH problem [33]: a DDH problem is solvable by means of a trapdoor (e.g., bilinear pairings) but, without it, the problem remains hard. In our proof, we show how a simulator tackles such a claim by solving the DDH problem without relying on any trapdoor. We model both hash functions H and $\mathcal{F}$ as random oracles such that the oracles $\mathcal{O}_H$ and $\mathcal{O}_{\mathcal{F}}$ are in charge of responding oracle queries for H and $\mathcal{F}$, respectively.

$\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Setup: $\mathcal{B}$ takes in a DDH challenge tuple $(g, g^a, g^b, z) \in \mathbb{G}_1^4$ of prime order p, as well as the description of $(\mathbb{G}_2, e)$, where $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. $\mathcal{B}$ sets the public key $pk = (p, \mathbb{G}_1, \mathbb{G}_2, g, e, v)$, where $v = z$, indicating that $\mathcal{B}$ implicitly sets sk by setting $g^{sk} = z$. $\mathcal{B}$ chooses a symmetric secret key $k \leftarrow$ **SE.gen**$(\mathcal{K})$ and sends pk to $\mathcal{A}$. Lastly, we put a restriction on the output of $\mathcal{O}_H$: we program the table of H such that $H(1) = g$.

Query: $\mathcal{A}$ makes **SigGen** queries adaptively: $\mathcal{A}$ selects a file $F = \{m_1, ..., m_n\}$ with identifier id and sends (F, id) to $\mathcal{B}$. Then, $\mathcal{B}$ encrypts F using k and generates σ. It sends (**SE.enc**$(F), \sigma)$ back to $\mathcal{A}$, where **SE.enc**(F) = (**SE.enc**$(m_1), ...,$ **SE.enc**$(m_n))$. $\mathcal{A}$ continues to query $\mathcal{B}$ for the ciphertexts **SE.enc**(F') and tags σ' on files F' and identifiers id' of its choice. $\mathcal{A}$'s queries are restricted such that $\mathcal{A}$ cannot make **SigGen** queries for two different files using the same identifier.

More precisely, $\mathcal{B}$ responds to $\mathcal{A}$'s queries as follows:

– If $\mathcal{A}$ makes a query for (F, id) that has not been made, then $\mathcal{B}$ computes **SE.enc**(F) = (**SE.enc**$(m_1), ...,$ **SE.enc**$(m_n))$. For $i \in \{1, ..., n\}$, $\mathcal{B}$ makes oracle queries for **SE.enc**(m_i) to $\mathcal{O}_{\mathcal{F}}$, which responds as follows:

(i) If $\mathcal{B}$ makes a query for **SE.enc**(m_i) that has not been made, then $\mathcal{O}_{\mathcal{F}}$ responds with $ct_i = \mathcal{F}($**SE.enc**$(m_i))$ and records the tuple (**SE.enc**$(F), ct_i)$.

(ii) If $\mathcal{B}$ made a query for the same tuple (**SE.enc**$(F), ct_i)$ previously, then $\mathcal{O}_{\mathcal{F}}$ retrieves the recorded value ct_i.

Next, for all $i \in \{1, ..., n\}$, $\mathcal{B}$ sets $s_i = \alpha^i \beta^i s$, where α, β are randomly chosen coefficients and s is a random variable. $\mathcal{B}$ makes oracle queries for id and $ct_i + s_i$ to $\mathcal{O}_H$, which responds as follows:

(i) If $\mathcal{B}$ makes a query for id and $ct_i + s_i$ that has not been made, then $\mathcal{O}_H$ responds with $H(id)$ and $H(ct_i + s_i)$, and records the tuple $(id, ct_i + s_i, H(id), H(ct_i + s_i))$.

(ii) If $\mathcal{B}$ made a query for the same tuple $(id, ct_i + s_i, H(id), H(ct_i + s_i))$ previously, then $\mathcal{O}_H$ retrieves the recorded values $H(id)$ and $H(ct_i + s_i)$.

Note that, due to the homomorphic property of H, the following equation holds:

$$\begin{aligned} H(id) \cdot H(ct_i + s_i) &= H(id) \cdot H(ct_i) \cdot H(s_i) \\ &= H(1 \cdot id) \cdot H(1 \cdot ct_i) \cdot H(1 \cdot s_i) \\ &= H(1)^{id} \cdot H(1)^{ct_i} \cdot H(1)^{s_i} \\ &= H(1)^{id + ct_i + s_i}. \end{aligned}$$

Since $\mathcal{O}_H$'s output in response to query "1" is g, we have

$$\begin{aligned} H(id) \cdot H(ct_i + s_i) &= g^{id} \cdot g^{ct_i} \cdot g^{s_i} \\ &= g^{id + ct_i + s_i}. \end{aligned}$$

$\mathcal{B}$ sets $\sigma_i = z^{id + ct_i + s_i}$ and returns **SE.enc**(F) and $\sigma = (\sigma_1, ..., \sigma_n)$. If $z = g^{ab} = g^{sk}$, then $\mathcal{B}$ simulates σ_i correctly. Finally, $\mathcal{B}$ records the following tuple

$$\begin{aligned} tup = (&F, \textbf{SE.enc}(F), id, \{ct_1, ..., ct_n\}, H(id), \\ &\{H(ct_1 + s_1), ..., H(ct_n + s_n)\}, \sigma). \end{aligned}$$

– If $\mathcal{A}$ made a query for the same (F, id) previously, then $\mathcal{B}$ retrieves the recorded tuple tup and returns **SE.enc**(F) and σ.

Challenge: $\mathcal{B}$ generates the challenge message $chal = \{(idx_i, \nu_i)\}_{idx_i \in I}$, where $I = \{idx_1, \ldots, idx_c\}$, idx_i are the indices of the blocks of file F of identity id, and $\nu_i \in_R \mathbb{Z}_p^*$. $\mathcal{B}$ sends $chal$ to $\mathcal{A}$.

Forge: $\mathcal{A}$ generates a proof $\{\sigma, \mu\}$ for the blocks $(m_{idx_1}, ..., m_{idx_c})$ of file F of identity id, determined by I. Then, it returns $\{\sigma, \mu\}$ to $\mathcal{B}$, and $\mathcal{B}$ checks the validity of $\{\sigma, \mu\}$. Since we assume that $\mathcal{A}$ wins the Auditing Game with non-negligible probability ϵ, $\{\sigma, \mu\}$ is a valid proof with the same probability. If **ProofVrfy**$(pk, \{\sigma, \mu\}, id, SI)$ returns $True$, then $\mathcal{B}$ returns 0, indicating that $z = g^{ab}$; otherwise $\mathcal{B}$ returns 1.

The proof further proceeds as follows. If $z = g^{ab}$, then $\mathcal{B}$ has simulated pk and $\mathcal{A}$'s **SigGen** queries correctly. As $\mathcal{A}$'s view in the simulation is identical to its view in the Auditing Game, $\mathcal{B}$'s probability to solve the DDH challenge tuple is reduced to $\mathcal{A}$'s probability to win the Auditing Game. Therefore, we have

$$\Pr[\mathcal{B}(g, g^a, g^b, z = g^{ab}) = 0] = \frac{1}{2} + \epsilon.$$

If z is a random group element, then $\mathcal{B}$ has not simulated pk and $\mathcal{A}$'s **SigGen** queries correctly. As $\mathcal{A}$'s view in the simulation is different from its view in the Auditing Game, $\mathcal{B}$ cannot exploit $\mathcal{A}$'s advantage, meaning that $\mathcal{B}$'s probability to determine if z is a random element is no better than flipping a coin. Therefore, we have

$$\Pr[\mathcal{B}(g, g^a, g^b, z = R) = 1] = \frac{1}{2},$$

where R is a random element in $\mathbb{G}_1$. This concludes the proof.

□

5.2 Performance Analysis

For our comparative analysis, we carefully chose the three proposals [12], [13], [40] as baseline schemes. Specifically, we selected Wang et al.'s [12] and Liu et al.'s [13] schemes which are seen as seminal works in terms of the public auditing of outsourced data. We additionally chose Guo et al.'s scheme [40], which we believe is the most relevant state-of-the-art proposal. We use these three schemes as a baseline to illustrate how efficiency can be improved while achieving the same level of security. We start by presenting a complexity analysis of the computation and communication costs. After that, we compare the actual performance of our scheme with the previous schemes in experiments.

TABLE 2: Computation costs

	Wang et al. [12]	Liu et al. [13].	Guo et al. [40]	Proposed scheme
KeyGen	$2\ Exp$	$1\ Exp$	$(c+1)\ Exp$	$1\ Exp$
SigGen	$n\ (Hash + Mult + 2\ Exp)$	$n\ (2\ Hash + Mult + Exp)$	$2n\ Exp$ $c\ MultExp$	$n\ (3\ Hash + Exp)$
ProofGen	$1\ Hash + c\ Add$ $+(c+1)\ Mult+$ $1\ Exp + c\ MultExp$	$1\ Hash + c\ Add$ $+c\ Mult + 1\ Exp$ $+c\ MultExp$	$c\ Exp$ $+2c\ Mult$	$c\ (Mult + Hash + Exp)$
ProofVrfy	$(c+1)\ (Hash + Mult)$ $+(c+2)\ Exp$ $+2\ Pair$	$(c+1)\ (Hash + Mult)$ $+(c+1)\ Exp$	$(c+2)\ Exp$ $+2\ pair$ $c\ Mult$	$(2c+1)\ Hash+$ $2\ Exp + 2\ Mult+$ $2Pair$

TABLE 3: Communication costs

	Wang et al. [12]	Liu et al. [13]	Guo et al. [40]	Proposed scheme
TPA $\leftrightarrow$ Cloud	$c \cdot (\lvert s\rvert + \lvert p\rvert + \lvert id\rvert) + 2\lvert p\rvert$	$c \cdot (\lvert s\rvert + \lvert p\rvert + \lvert id\rvert) + 2\lvert p\rvert$	$c \cdot (3 \cdot \lvert p\rvert + \lvert g\rvert)$	$c \cdot (\lvert s\rvert + \lvert p\rvert) + \lvert g\rvert + \lvert p\rvert$

5.2.1 Computation Costs

The computation costs for the four schemes are summarized in Table 2. Let $Hash$ be the hash operation, Add be the additive operation in group $\mathbb{Z}_p$, $Mult$ be the multiplicative operation in group $\mathbb{Z}_p$, Exp be the exponential operation in group $\mathbb{G}_1$, Mod be the mod operation in group $\mathbb{Z}_p$, $MultExp$ be the multiplication sequence for the exponential operation in group $\mathbb{G}_1$, and $Pair$ be the pairing operation. We omit non-dominant operations such as running a symmetric-key encryption algorithm. Suppose there are c random blocks specified in $chal$ during the auditing process. Under this setting, we provide a calculation of the computation costs for four algorithms: **KeyGen SigGen**, **ProofGen**, **Challenge**, and **ProofVrfy**.

In the **KeyGen** algorithm, only Guo et al.'s scheme shows linear complexity with respect to c in the key generation phase, while the other schemes require a constant number of operations. In the **SigGen** algorithm, the proposed scheme has a higher computation complexity than Wang et al.'s [12] and Liu et al.'s [13] schemes. This is because our scheme uses a symmetric-key encryption algorithm to ensure data confidentiality, which incurs additional hash operations. Lastly, Guo et al.'s scheme exhibits the worst computation complexity, but this is compensated for by achieving $\mathcal{O}(\log n)$ update costs. In the **ProofGen** algorithm, the computation cost of the proposed scheme is lower than [12], [13], [40]. For the **ProofVrfy** algorithm, all of the schemes show similar performance except for [12].

5.2.2 Communication Costs

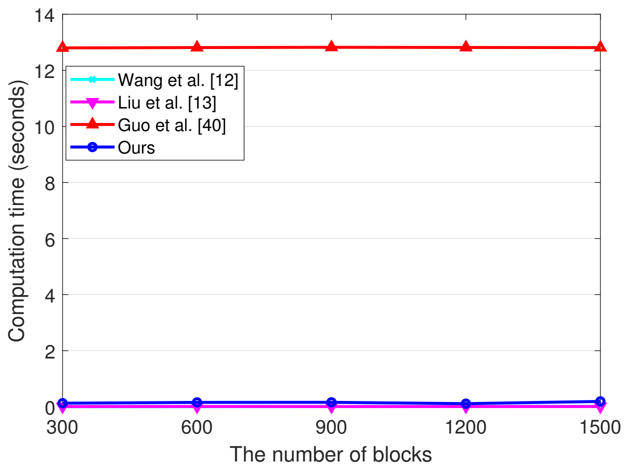
Table 3 shows a comparison of the communication costs for the four schemes. The size of the challenge message $chal$=$\{(idx_i, \nu_i)\}_{idx_i \in I}$ is $c \cdot (|s| + |p|)$ bits, where c is the number of selected blocks, $|s|$ is the size of each index in $\{idx_1, \ldots, idx_c\}$, and $|p|$ is the size of an element of $\mathbb{Z}_p$. Let $|g|$ be the size of an element of $\mathbb{G}_1$. The size of $proof$ is $|p| + |g|$ bits.

In [12], [13], the cloud uses a random mask to guarantee data privacy against the TPA, where the size of random mask R is $|p|$. Owing to the use of side information, the communication cost between the cloud or the TPA is $c \cdot |p|$. Guo et al.'s scheme transfers elements in $\mathbb{G}$ during the auditing phase, where these values correspond to the tags [40], while the proposed scheme does the same except that the tags are aggregated into a single value.

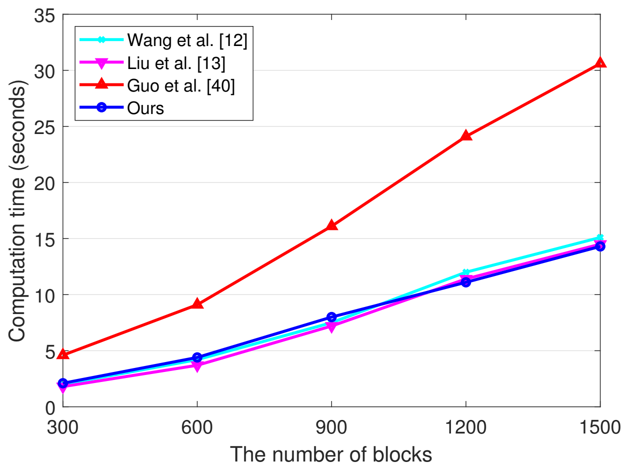
5.2.3 Simulation

We measured the computation time for four algorithms: **KeyGen**, **SigGen**, **ProofGen**, and **ProofVrfy**. Because the computation costs for the challenge algorithm are the same for all schemes, we excluded this from the comparison. The experiment was conducted using C in Linux Ubuntu 3.19.0-15-generic OS 64-bit (Workstation 11.0 virtual machine, 1 GB of RAM) with an Intel Core i5 processor running at 2.3 GHz with 8 GB of RAM. We utilized the Pairing Based Cryptography (PBC) library (version 0.5.14 [29]) to implement the pairing operation in our scheme. The experiment was performed on five randomly generated files, where each file consists of $n \in \{300, 600, 900, 1200, 1500\}$ blocks. The size of each block was set at 20 bytes, and the seed length $l(\lambda)$ was set at half of the size of the block. For proof generation and verification, the performance depends on c, the number of challenged blocks. Note that 300∼460 challenged blocks are sufficient for successful verification with a probability of 95∼99%, irrespective of the number of blocks a file consists of [5]. In our experiment, we set $c = 300$.

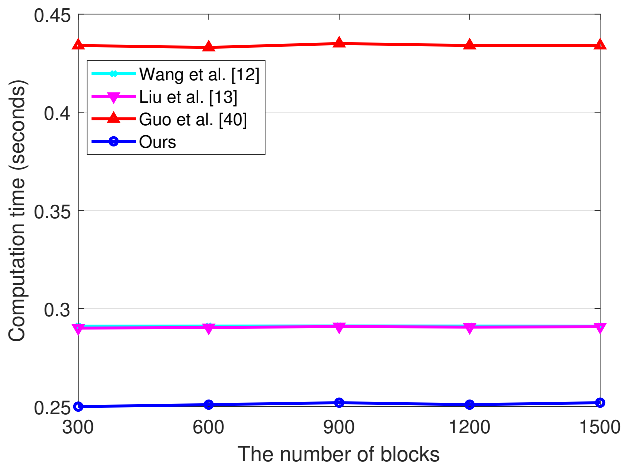
Figure 4 demonstrates the time costs for public auditing for the four schemes. Figure 4(a) shows the computation time for the **KeyGen** algorithm. As we showed in the theoretical analysis, Wang et al.'s [12], Liu et al.'s [13] and our schemes exhibit similar performance. Guo et al.'s scheme shows the worst performance because its computation complexity with respect to key generation depends on c [40]. Figure 4(b) presents the computation time for the **SigGen** algorithm. The proposed scheme is much faster than [40], and shows similar performance to [12], [13]. Figure 4(c) shows the computation time for the **ProofGen** algorithm. In this phase, the proposed scheme does not use random masking, so it is slightly faster than [12], [13] by about 0.18



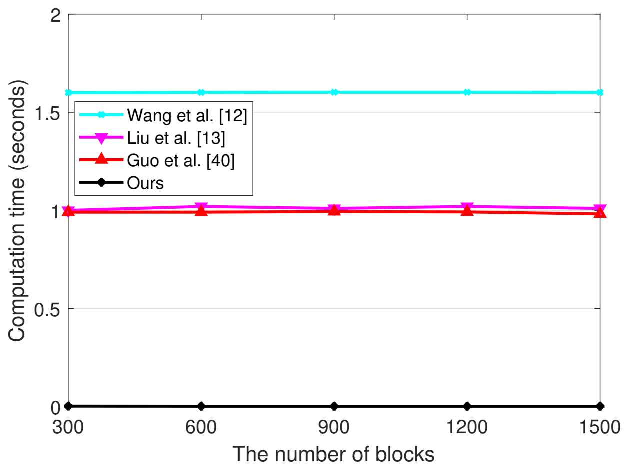
(a) Key generation



(b) Tag generation



(c) Proof generation



(d) Proof verification

Fig. 4: Computation costs of public auditing in the various schemes

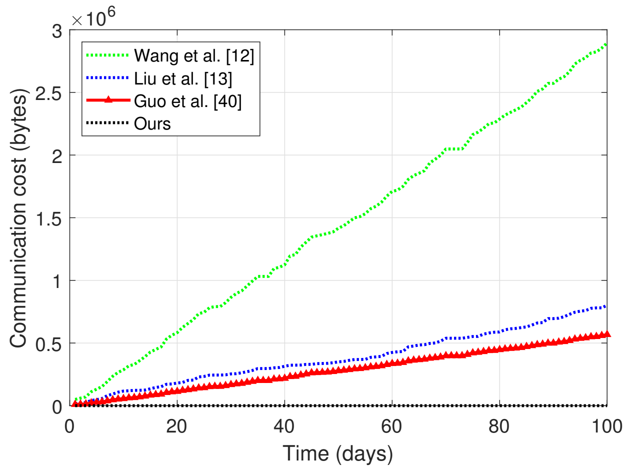


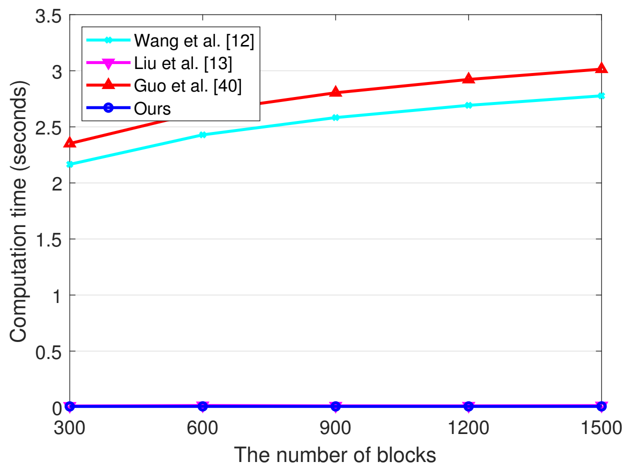
Fig. 5: Communication costs for data dynamics in the various schemes

seconds. Moreover, the performance of the proposed scheme surpasses that of [40].

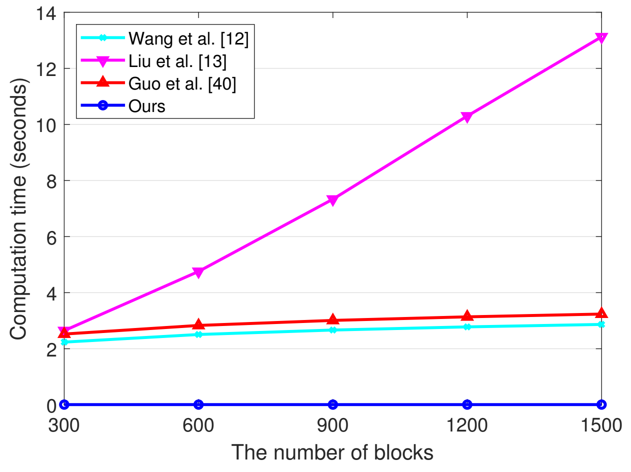
Figure 4(d) displays the computation time for the **ProofVrfy** algorithm. Ours exhibits the best performance while [13] and [40] show similar performance. The performance of [12] is the worst due to its greater number of exponential computations. Note that the proposed scheme requires approximately 4 ms on average, which is much faster than others. If we use pre-computation, then the proof verification time is reduced to 2 ms on average. The downside of pre-computation is the storage overhead required for each file. Specifically, the TPA is required to keep two pre-computed values, each of which is a 128-byte element in $\mathbb{G}_1$. This cost, however, is acceptable. For example, when simultaneously auditing numerous files, say one million, only 256 MB is required, which is tolerable for modern server systems.

Next, we measure the communication costs of the auditing schemes when supporting data dynamics. We consider an online cloud server connected to the Internet. It has been shown that file transfer can be accurately modeled as a Poisson process [38]. Because the number of file transfers follows a Poisson distribution with rate λ, we present the simulation results for this probabilistic behavior distribution.

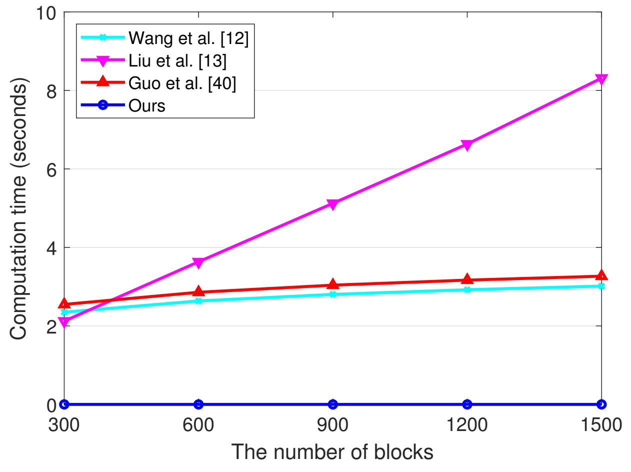
Figure 5 presents the accumulated communication costs when a user requests data updates from the public auditing



(a) Modification



(b) Insertion



(c) Deletion

Fig. 6: Computation cost of dynamic update

schemes over 100 days. In this simulation, we assume that a user transfers a request for dynamic operations on a file approximately three times a day ($\lambda = 3$). The number of blocks per file can vary in real-world systems, so we assume that every file consists of an arbitrary number of blocks, ranging from 100 to 90,000. The communication costs, which are measured in bytes, include all of the data relating to auditing, such as newly generated tags and auditing side information. In Figure 5, the proposed scheme shows the best performance because all of the other schemes, for the update of a single i-th block, should transfer newly generated tags corresponding to the j-th blocks ($j \geq i$).

Figure 6 depicts the computation costs with respect to dynamic updates. To assess the performance of each scheme in the worst case scenario, we assume that the first block of an n-block file is updated (i.e., modification, insertion, and deletion). In the proposed scheme, modification and insertion (Figure 6(a) and 6(b)) has constant computation costs (0.007 seconds on average to compute a modified (or newly inserted) tag) irrespective of the number of blocks. For deletion, it does not incur any cost with respect to tag generation. In contrast, modification, insertion, and deletion incur logarithmic costs in [13], [40]. This is because all of the remaining tags corresponding to the i-th blocks ($i \geq 2$) in a tree structure need to be recomputed. Similarly, when insertion or deletion at the first block occurs, Liu et al.'s scheme needs to compute $n - 1$ tags, corresponding to the remaining blocks [17].

6 Conclusion

In this paper, we propose a public auditing scheme for encrypted data that supports extremely fast data dynamics. The proposed scheme supports data dynamics at a constant cost irrespective of the number of blocks. Our auditing challenge-response protocol requires a constant number of pairings and exponentiations, which significantly increases the verification speed for the auditing results. The proposed scheme ensures data confidentiality and integrity against the cloud server. During the auditing process, the TPA can verify the correctness of the proof without decrypting it and without key exposure, due to the homomorphic hash function. Security and performance analysis shows that the proposed scheme requires minimal extra computation while guaranteeing data privacy and integrity.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00533, Research on CPU vulnerability detection and validation). This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ICT Creative Consilience program(IITP-2020-0-01819) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).

References

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.
[2] Ren, K., Wang, C., & Wang, Q. (2012). Security challenges for the public cloud. IEEE Internet Computing, 16(1), 69-73.
[3] Song, D., Shi, E., Fischer, I., & Shankar, U. (2012). Cloud data protection for the masses. Computer, 45(1), 39-45.
[4] Wei, L., Zhu, H., Cao, Z., Dong, X., Jia, W., Chen, Y., & Vasilakos, A. V. (2014). Security and privacy for storage and computation in cloud computing. Information sciences, 258, 371-386.

[5] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., & Song, D. (2007, October). Provable data possession at untrusted stores. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 598-609).

[6] Ateniese, G., Di Pietro, R., Mancini, L. V., & Tsudik, G. (2008, September). Scalable and efficient provable data possession. In Proceedings of the 4th international conference on Security and privacy in communication netowrks (pp. 1-10).

[7] Juels, A., & Kaliski Jr, B. S. (2007, October). PORs: Proofs of retrievability for large files. In Proceedings of the 14th ACM conference on Computer and communications security (pp. 584-597).

[8] Shacham, H., & Waters, B. (2008, December). Compact proofs of retrievability. In International conference on the theory and application of cryptology and information security (pp. 90-107). Springer, Berlin, Heidelberg.

[9] Erway, C. C., Küpçü, A., Papamanthou, C., & Tamassia, R. (2015). Dynamic provable data possession. ACM Transactions on Information and System Security (TISSEC), 17(4), 1-29.

[10] Wang, Q., Wang, C., Ren, K., Lou, W., & Li, J. (2010). Enabling public auditability and data dynamics for storage security in cloud computing. IEEE transactions on parallel and distributed systems, 22(5), 847-859.

[11] Zhu, Y., Wang, H., Hu, Z., Ahn, G. J., Hu, H., & Yau, S. S. (2011, March). Dynamic audit services for integrity verification of outsourced storages in clouds. In Proceedings of the 2011 ACM Symposium on Applied Computing (pp. 1550-1557).

[12] Wang, C., Chow, S. S., Wang, Q., Ren, K., & Lou, W. (2011). Privacy-preserving public auditing for secure cloud storage. IEEE transactions on computers, 62(2), 362-375.

[13] Liu, H., Zhang, P., & Liu, J. (2013). Public data integrity verification for secure cloud storage. Journal of networks, 8(2), 373.

[14] Shen, J., Shen, J., Chen, X., Huang, X., & Susilo, W. (2017). An efficient public auditing protocol with novel dynamic structure for cloud data. IEEE Transactions on Information Forensics and Security, 12(10), 2402-2415.

[15] Shah, M. A., Swaminathan, R., & Baker, M. (2008). Privacy-Preserving Audit and Extraction of Digital Contents. IACR Cryptol. ePrint Arch., 2008, 186.

[16] Liu, J., Huang, K., Rong, H., Wang, H., & Xian, M. (2015). Privacy-preserving public auditing for regenerating-code-based cloud storage. IEEE transactions on information forensics and security, 10(7), 1513-1528.

[17] Ramaiah, Y. G., & Kumari, G. V. (2013, July). Complete privacy preserving auditing for data integrity in cloud computing. In 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (pp. 1559-1566). IEEE.

[18] Wang, J., Chen, X., Huang, X., You, I., & Xiang, Y. (2015). Verifiable auditing for outsourced database in cloud computing. IEEE transactions on computers, 64(11), 3293-3303.

[19] Jin, H., Jiang, H., & Zhou, K. (2016). Dynamic and public auditing with fair arbitration for cloud data. IEEE Transactions on cloud computing, 6(3), 680-693.

[20] Tian, H., Chen, Y., Chang, C. C., Jiang, H., Huang, Y., Chen, Y., & Liu, J. (2015). Dynamic-hash-table based public auditing for secure cloud storage. IEEE Transactions on Services Computing, 10(5), 701-714.

[21] Naone, E. (2010). What Twitter learns from all those tweets. Technology Review, 28.

[22] Krohn, M. N., Freedman, M. J., & Mazieres, D. (2004, May). On-the-fly verification of rateless erasure codes for efficient content distribution. In IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004 (pp. 226-240). IEEE.

[23] Gennaro, R., Katz, J., Krawczyk, H., & Rabin, T. (2010, May). Secure network coding over the integers. In International Workshop on Public Key Cryptography (pp. 142-160). Springer, Berlin, Heidelberg.

[24] Boneh, D., Lynn, B., & Shacham, H. (2001, December). Short signatures from the Weil pairing. In International conference on the theory and application of cryptology and information security (pp. 514-532). Springer, Berlin, Heidelberg.

[25] Xia, H., Lu, T., Shao, B., Ding, X., & Gu, N. (2014, May). Hermes: On collaboration across heterogeneous collaborative editing services in the cloud. In Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD) (pp. 655–660). IEEE.

[26] Bao, F., Deng, R. H., & Zhu, H. (2003, October). Variations of diffie-hellman problem. In International conference on information and communications security (pp. 301-312). Springer, Berlin, Heidelberg.

[27] Armknecht, F., Bohli, J. M., Karame, G. O., Liu, Z., & Reuter, C. A. (2014, November). Outsourced proofs of retrievability. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (pp. 831-843).

[28] Zhang, Y., Xu, C., Cheng, N., Li, H., Yang, H., & Shen, X. (2019). Chronos $^{+}$: An Accurate Blockchain-Based Time-Stamping Scheme for Cloud Storage. IEEE Transactions on Services Computing, 13(2), 216-229.

[29] Lynn, B. (2006). The pairing-based cryptography library. Internet: crypto. stanford. edu/pbc/[Mar. 27, 2013].

[30] McKeen, F., Alexandrovich, I., Berenzon, A., Rozas, C. V., Shafi, H., Shanbhogue, V., & Savagaonkar, U. R. (2013). Innovative instructions and software model for isolated execution. Hasp@ isca, 10(1).

[31] Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., ... & Stoica, I. (2009). Above the clouds: A berkeley view of cloud computing. Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS, 28(13), 2009.

[32] Zhang, X., Wang, H., & Xu, C. (2019). Identity-based key-exposure resilient cloud storage public auditing scheme from lattices. Information Sciences, 472, 223-234.

[33] Dent, A. W. & Galbraith, S. D. (2006, July). Hidden pairings and trapdoor DDH groups. In International Algorithmic Number Theory Symposium (pp. 436-451). Springer, Berlin, Heidelberg.

[34] Fan, K., Bao, Z., Liu, M., Vasilakos, A. V., & Shi, W. (2020). Dredas: Decentralized, reliable and efficient remote outsourced data auditing scheme with blockchain smart contract for industrial IoT. Future Generation Computer Systems, 110, 665-674.

[35] Rabaninejad, R., Ahmadian, M., Asaar, M. R., & reza Aref, M. (2019). A lightweight auditing service for shared data with secure user revocation in cloud storage. IEEE Transactions on Services Computing.

[36] Zhang, Y., Xu, C., Lin, X., & Shen, X. S. (2019). Blockchain-based public integrity verification for cloud storage against procrastinating auditors. IEEE Transactions on Cloud Computing.

[37] Wang, H., Wang, Q., & He, D. (2019). Blockchain-Based Private Provable Data Possession. IEEE Transactions on Dependable and Secure Computing.

[38] Carofiglio, G., Gallo, M., Muscariello, L., & Perino, D. (2011, September). Modeling data transfer in content-centric networking. In Proceedings of the 23rd international teletraffic congress (pp. 111–118). International Teletraffic Congress.

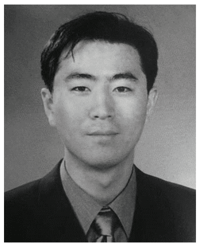
[39] Erway, C. C., Küpçü, A., Papamanthou, C., & Tamassia, R. (2015). Dynamic provable data possession. ACM Transactions on Information and System Security (TISSEC), 17(4), 15.

[40] Guo, W., Zhang, H., Qin, S., Gao, F., Jin, Z., Li, W., & Wen, Q. (2019). Outsourced dynamic provable data possession with batch update for secure cloud storage. Future Generation Computer Systems, 95, 309-322.

Changhee Hahn received B.S. and M.S. degrees from Chung-Ang University, Seoul, South Korea, in 2014 and 2016, respectively, both in Computer Science. He received the Ph.D. degree in 2020 in the Department of Computer Science and Engineering, College of Informatics, Korea University, Korea. His research interests include information security and cloud computing security.

Hyunsoo Kwon received a B.S. degree from Chung-Ang University, Seoul, Korea, in 2014 and a M.S. degree from Korea University, Seoul, South Korea, in 2016, both in Computer Science. He is currently pursuing a Ph.D. degree in the Department of Computer Science and Engineering, College of Informatics, Korea University, Korea. His research interests include information security, network security, and cloud security.

Daeyeong Kim received a B.S. in Computer Science and Engineering from Dae-jeon University, Daejeon, Korea, in 2014. He is currently pursuing an M.S. degree in the Department of Computer Science and Engineering, College of Informatics, Korea University, Korea. His research interests include information security and cloud security.



Junbeom Hur received a B.S. degree from Korea University, Seoul, South Korea, in 2001, and M.S. and Ph.D. degrees from KAIST in 2005 and 2009, respectively, all in Computer Science. He was with the University of Illinois at Urbana-Champaign as a postdoctoral researcher from 2009 to 2011. He was with the School of Computer Science and Engineering at the Chung-Ang University, South Korea as an Assistant Professor from 2011 to 2015. He is currently an Associate Professor with the Department of Computer Science and Engineering at Korea University, South Korea. His research interests include information security, cloud computing security, mobile security, and applied cryptography.