IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 22, 2019, accepted May 13, 2019, date of publication May 17, 2019, date of current version June 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917628

Cryptographic Accumulator-Based Scheme for Critical Data Integrity Verification in Cloud Storage

WALID I. KHEDR, HEBA M. KHATER, AND EHAB R. MOHAMED
Faculty of Computers & Informatics, Zagazig University, Zagazig 44519, Egypt
Corresponding author: Walid I. Khedr (wkhedr@fci.zu.edu.eg)

ABSTRACT Public cloud storage is a fundamental cloud computing service. Currently, most owners of large data outsource their data to cloud storage services—even high-profile owners such as governments. However, public cloud storage services are not optimal for ensuring the possession and integrity of the outsourced data, a situation that has given rise to many proposed provable data possession check schemes (PDP). A PDP scheme allows data owners to efficiently, periodically, and securely verify that a cloud storage provider possesses the outsourced data. Most of the currently available provable data possession check schemes make selective (i.e., probabilistic) checks using random data blocks to verify data integrity rather than checking the entire dataset. Therefore, these schemes are considered inadequate by critical infrastructure sectors that involve highly sensitive data (critical data). In this paper, a new and efficient deterministic data integrity check scheme called cryptographic-accumulator provable data possession (CAPDP) is proposed. The CAPDP surpasses the common limitations exhibited by other currently proposed schemes. The underlying scheme of the CAPDP is based on a modified RSA-based cryptographic accumulator that has the following advantages: 1) it allows the data owner to perform an unlimited number of data integrity checks; 2) it supports data dynamics; 3) it is efficient in terms of communication, computation and storage costs for both the data owner and the cloud storage provider; 4) the verification operation in the proposed scheme is independent of the number of blocks being verified; 5) it minimizes the burden and cost of the verification process on the data owner's side, enabling verification to be performed even on low-power devices; and 6) it prevents tag forgery, data deletion, replacement, and data leakage attacks and detects replay attacks. Moreover, the prototype implementation and experimental results show that the scheme is applicable in real-life applications.

INDEX TERMS Cloud storage, cryptographic accumulator, data integrity verification, dynamic operations.

I. INTRODUCTION

Cloud computing is based on outsourcing computing resources rather than relying on local servers or personal devices [1]. Outsourcing large amounts of data from different devices to a cloud storage provider (CSP) saves data owners from having to acquire a concomitant amount of storage without compromising their ability to manage these data remotely. Nevertheless, data owners worry about the possibility of uploaded data being modified or erased without their knowledge or permission. To resolve this concern, a data owner must execute regular data integrity checks to verify data possession and detect any unauthorized modifications.

The associate editor coordinating the review of this manuscript and approving it for publication was Yinghui Zhang.

Provable data possession (PDP) is a technique for verifying the integrity of the outsourced data. The main objective of any PDP scheme is

to enable a verifier to efficiently, periodically and securely verify that a cloud storage provider is not deceiving the data owner. Technically, there are two approaches for performing such integrity verification [2]:

1) The deterministic approach: In this approach, the data owner requests that all the data blocks be checked to verify their integrity, thus providing a 100% guarantee of the integrity and possession of the data.
2) The probabilistic approach: In this approach, the data owner requests random checks of chosen blocks of data to verify their integrity, thus providing less than a 100% guarantee of the integrity and possession of the data.

As reported in [3]–[5] cloud computing has been adopted by critical infrastructure sectors such as energy, finance, transport, and many others. In the U.S., for example, the C.I.A. recently partnered with Amazon web services [6] as its cloud storage provider [5], whereas the U.S. Departments of State, Homeland Security, and Veterans Affairs have partnered with NetApp [7], [8]. All these federal agencies have strict requirements regarding data integrity and possession and will accept no less than a 100% guarantee of intact data because if any of their files were damaged or corrupted, they could potentially suffer a huge loss. Likewise, if any of their confidential files were abused, that would be a disaster. Allowing any party to verify the integrity of the infrastructure sectors' critical data may allow that party to obtain useful information about the sensitive files. Therefore, we propose a deterministic private verification integrity check scheme to avoid this situation while providing a 100% guarantee that the data are correct. The deterministic nature of this scheme means it is immune to CSP data falsification because it checks on all the data blocks in a dataset rather than only a subset. Moreover, because this scheme is private, the data owners can feel confident about the safety of their data; i.e., no data leakage will occur. In practice, however, a CSP using the deterministic approach would incur high computation overhead, especially when verifying sizeable datasets [2], [9]. Thus, another limitation to the approach is the limited number of data integrity checks a data owner can perform. To overcome these limitations, our deterministic scheme uses a modified RSA-based cryptographic accumulator [10]. The main contributions of the proposed scheme are:

1) The data owner would be able to execute an unlimited number of data verifications against the entire dataset while limiting computational overhead to acceptable levels.
2) The proposed scheme supports features such as blockless verification and dynamic data operations.
3) In comparison to other deterministic schemes, the proposed scheme is highly secure against tag forgery, data deletion, replacement, and data leakage attacks.
4) The verification operation in the proposed scheme is independent of the number of blocks being verified.
5) The proposed scheme minimizes the burden and cost of the verification process on the data owner's side, enabling verification to be performed even on low-power devices
6) The proposed scheme is efficient in terms of communication, computation and storage costs, for both the data owner and the CSP.

Prototype implementation and experimental results using the proposed scheme show that it is applicable in real-life applications.

A. RELATED WORK

As mentioned above, data integrity checks are classified into two approaches: probabilistic and deterministic. Although a wide variety of probabilistic schemes have been proposed [11]–[13], few deterministic schemes have been proposed due to the limitations mentioned above. In his work, Caronni and Waldvogel [14] presented a simple deterministic mechanism that supports public verifiability of the data but not the handling of dynamic operations. A data owner using his scheme would not be able to insert, modify and/or delete any of the data uploaded to the provider. Every time the data owner would send a challenge message to the provider, she would receive a Message Authentication Code (MAC) of the data in response. To cross-check the data's authenticity, therefore, the data owner would need to have a replicated copy of the data. Another scheme was proposed by Deswarte *et al.* [15] based on the well-known cryptographic protocol of Diffie-Hellman for key exchange. Two main limitations were reported in this scheme. First, the CSP must compute exponentiation over the entire file in each verification process which results in high computation overhead for the CSP, especially when dealing with large files. In addition, the number of challenges allowed for a data owner to check the data integrity is limited. To overcome the second limitation, Filho and Barreto [16] proposed a simple deterministic data integrity check protocol that is based on an RSA-based homomorphic hash function. Although the number of verifications in his scheme was limitless, the computation cost remained high and the data dynamics were not supported as well. Sebé *et al.* [17] then proposed a protocol based on the Diffie-Hellman key exchange that did not require computing the exponentiation over the entire file. It divided a data file into blocks and generated an RSA-based homomorphic hash function for each block. Using this scheme, the computation cost for the CSP decreased but caused the data owners storage cost to increase due to the large size of the hash values that were stored for each block. Another drawback of the scheme was that it neither supported data dynamic operations nor public integrity checks. Following the work of Sebe *et al.,* Barsoum [18] proposed a multicopy provable data possession scheme to support the public verifiability of multiple replicas of the data while preserving the data privacy. To support all; the dynamic data operations, the public verifiability, and the data privacy, Hao *et al.* and Yi *et al.* proposed remote data integrity check schemes [19], [20]. A verifier using Hao *et al.* scheme would need a long time to setup and verify the data due to the large number of modular exponentiations. Furthermore, their scheme did not prevent any leakage of the data from malicious providers. Regarding Yi *et al.* scheme, they proposed a probabilistic/deterministic multicopy Provable Data Possession (PDP) scheme which is based on Fully Homomorphic Encryption (FHE) algorithm [21]. It verifies the integrity of replicated data stored on multiple servers across multiple data centers. The authors have used a third-party verification approach for analyzing the file copies according to the reliability and integrity conditions based on the verification methods. However, the scheme is inefficient in terms of communication cost as it requires the CSP to

send all the challenged blocks of each data to a third-party auditor (TPA) for verification.

B. ORGANIZATION OF THE PAPER

The rest of this paper is organized as follows: Section II describes the preliminaries for the proposed scheme. Section III describes the structure of the proposed scheme. Section IV elaborates the scheme's characteristics. Sections V and VI discuss the security and performance analysis respectively. Section VII concludes this paper.

II. PRELIMINARIES

In this Section, we will provide a short note concerning the concepts used throughout the paper.

A. THE RSA ONE-WAY ACCUMULATOR

One-way accumulators were first proposed by Benaloh and de Mare [22] and are defined as one-way hash functions with the quasi-commutative property. A one-way hash function $h : X \times Y \rightarrow X$ satisfies the quasi-commutative property, if for all $x \in X$ and for all $y_1, y_2 \in Y$:

$$h(h(x, y_1), y_2) = h(h(x, y_2), y_1) \tag{1}$$

One-way accumulators can be useful in multiple situations. First, they can accumulate a finite set $X = \{x_1, \cdots, x_n\}$ into a secure digest called an accumulator, acc_X, by repeatedly applying h to each x_i. Note that acc_X does not depend on the order in which the x_i values are accumulated. Considering $g \in G$ as an initial value, the one-way accumulator is defined as following [22]:

$$acc_X = h(h(h(\cdots h(h(h(g, x_1), x_2), x_3), \cdots, x_{n-2}), x_{n-1}), x_n)$$

Second, these accumulators can also be used to generate a witness w_j to verify the membership of an element x_j in X by accumulating all the elements x_i, such that $i \neq j$. Then, the membership of x_j in X is verified if $h(w_j, x_j) = acc_X$. It should be computationally infeasible for a probabilistic polynomial-time adversary to find a witness w' for any value $x' \notin X$ such that $h(w', x') = acc_X$ due to the collision-freeness property [23]. One well-known implementation of one-way accumulators is the RSA accumulator [22].

The basic RSA accumulator works as follows. Given that two strong primes [24] p and q are t-bit integers such that $N = pq$ and given that a base $g \in \mathbb{Z}_N$ is coprime to N, the accumulation value for a set $X = \{x_1, \cdots, x_n\}$ is computed as shown below:

$$acc_X = g^{x_1 \cdots x_n} \bmod N. \tag{2}$$

The modulus N should be at least k-bits, where k is the number of bits required for the maximum number in the set X [10]. To verify the membership of x_j in X, a witness w_j is generated by accumulating all the elements in X except x_j:

$$w_j = g^{x_1 \cdots x_{j-1} x_{j+1} \cdots x_n} \bmod N. \tag{3}$$

The membership of x_j in X is verified if $w_j^{x_j} \equiv acc_X \bmod N$. The elements of the set X must be restricted to only prime numbers to avoid collision [23], i.e., it should be computationally infeasible to generate a witness for an unaccumulated element. Because most applications must accumulate arbitrary integers, a prime representative [25] for each $x_i \in X$ is usually generated and used as an input to the RSA accumulator. However, different variants of RSA accumulators that relax the primality constraint have been proposed [26]–[28]. In this paper, we propose a new algorithm (BlockGen) that eliminates the need for the primality constraint while avoiding collisions, as discussed in Section III-A-1.

B. CHARACTERISTICS OF CLOUD DATA INTEGRITY SCHEMES

All cloud data integrity schemes should possess all or a subset of the following characteristics [2]:

C1) **Block-less verification** means that the data owner should not have to retrieve the complete file blocks from the cloud provider for verification.

C2) **Unrestricted challenge frequency** means that the data owner is not restricted to a limited number of challenges when verifying the integrity of the outsourced data.

C3) **Soundness:** This means that the CSP cannot pass a challenge request without actually holding the data or with corrupted data.

C4) **Stateless verification:** Every challenge request is independent of all previous verifications with respect to the CSP and the data owner.

C5) **Robustness:** is the ability to identify even insignificant corruptions of the data.

C6) **Data recovery:** is the ability to support data recovery along with identifying any data corruptions.

C7) **Dynamic data handling:** indicates that the dynamic data (insertion, deletion, and modification) should be supported by the integrity scheme.

C8) **Public auditability:** The data owner can delegate the task of the data integrity check to a third-party auditor (TPA).

C9) **Privacy-preserving:** A third party auditor can verify the integrity of the outsourced data but cannot breach the confidentiality of the data.

C10) **Fairness** means that dishonest users cannot accuse a reliable CSP of tampering with the outsourced data.

C. OVERVIEW OF THE PROPOSED CAPDP SCHEME

Two main roles exist in any cloud storage service: (1) the cloud storage provider (CSP), whose main objective is to offer the required storage space to data owners who wish to outsource their data; and (2) the data owner, who takes a risk by saving significant amounts of critical data in the CSP without having backup copies and demands a 100% data integrity and possession guarantee due to the importance of the data outsourced. In this paper, we hold the realistic assumption that the connection between these two players is secured against all types of attacks through the use of a secure socket layer (SSL) connection. We also assume that the CSP has the computational resources required to perform any cryptographic operations on the outsourced data and that

TABLE 1. Notations.

Symbol	Meaning
C	The data owner
CSP	The cloud storage provider
k_E	A symmetric key that is known only to the data owner
k_H	A symmetric key that is known only to the data owner
$H(\cdot)$	A secure one-way hash function
$Cert_C$	Data owner's certificate
PR_C	Data owner's private key
PR_{TG}	Private key used for the tag generation process
$Cert_{TG}$	Data owner's tag generation certificate
TS	Timestamp
$E_k(\cdot)$	Symmetric/Asymmetric key encryption with key k
$D_k(\cdot)$	Symmetric/ Asymmetric key decryption with key k
M	The outsourced Data
b_i''	Data block
acc_B	RSA accumulator
w_j	Witness
$Accr$	Accumulator-based register
τ	A data block tag
$\|x\|$	The size of x

it may behave unreliably while doing so. Based on these assumptions, we carefully designed our deterministic scheme to ensure that the CSP has the complete version of the data to be able to pass a verification process successfully. Table 1 shows a list of notations used throughout the paper. The main idea underlying the proposed scheme is that the data M is split into n encrypted segments of ℓ_1-bits each using k_E, $M = \{m_1, \cdots m_n\}$. A block generation (BlockGen) algorithm is used to generate a ℓ_2-bit tag τ_i for each segment m_i and store it at the data owner's side. A data block b_i'' is then generated by using each tag concatenated to its corresponding segment, such that $B'' = \{b_1'', \cdots, b_n''\}$. The data owner C uses (2) to compute the accumulation value $acc_{B''}$ for the set B'' and then outsources B'' to the CSP. To verify the integrity of M, the data owner challenges the CSP with a random block index j. In turn, the CSP uses the proof generation (ProofGen) algorithm to compute the witness w_j for block b_j'' by accumulating all the elements in B'' except b_j'' using (3). Note that the ProofGen algorithm compels the CSP to use all the data blocks except b_j'' to compute w_j. The CSP then returns w_j and b_j'' to the data owner, who uses them to execute the block verification algorithm (ProofVer) to check the data integrity. All three algorithms, BlockGen, ProofGen, and ProofVer, use this basic CAPDP scheme for static data and are discussed in Section III-A. For the dynamic data support discussed in Section III-B, a new data structure called a tag record table (TRT) is introduced.

D. TAG RECORD TABLE

The TRT data structure is a simplified version of the map-version table (MVT) introduced in [29]. This table tracks data blocks during all dynamic operations by storing the tags (τ) that uniquely identify each block at the data owner's side. It is composed of two columns: the block index (BI), which represents the real-time index of the block, and the tag value (TV), which represents the block tag (τ_i) that uniquely identifies the block. In addition, the TRT is used to sort data blocks when retrieving data files from the CSP. Although this approach demands extra storage space from the data owner (due to the storage overhead of the TRT), the overall space required is small. For example, a 1 GB file with a 384-byte block size would require 2% of the file's size space to store the TRT.

E. SECURITY REQUIREMENTS

Because the CSP is not fully trusted, it could potentially behave maliciously regarding the outsourced data and hide any damage that occurred in the data. According to [2], secure data integrity check schemes should resist the following attacks that are launched by a semi-trusted CSP:

S1) ***Tag forgery attack***: Almost all cloud data integrity check schemes use tags, which are metadata appended to the original data. To bypass an integrity verification process, the CSP may try to forge these tags.

S2) ***Data deletion attack***: CSP may use only the tags to generate valid proof of data possession even when the original data may be corrupted or deleted.

S3) ***Replace attack***: In response to a data possession challenge, the CSP may replace the corrupted or deleted data blocks and their corresponding tags with other valid data blocks and tags.

S4) ***Replay attack***: CSP might use an old challenge response to respond to a new challenge that matches it

S5) ***Data leakage attack***: An attacker might reveal stored data during the verification process.

A secure RDPC protocol that resists all the attacks mentioned above guarantees that the CSP cannot successfully pass the verification process without possessing all the data blocks. Referring to [30], we initiate a data possession checking game to prove that our proposed integrity scheme is secure against all the above-mentioned attacks. The game includes a challenger C and an adversary A that serves as a semi-trusted CSP. The game is conducted as follows:

Setup: C generates two symmetric keys k_H and k_E.

Query: A selects some file segments of its choice m_i, where $1 \leq i \leq n$, and sends it to C. C executes the BlockGen algorithm to obtain a valid tag (verification metadata) for each segment and then returns all the computed tags τ concatenated to the encrypted file segments $M = \{m_1, \cdots m_n\}$ as $b_i = m_i||\tau_i$ to A. A continues to query C to generate tags on the segments of its choice and stores the results on its side.

Challenge: C challenges A with a random block index j.

Forge: A computes the proof P and sends it as a valid proof to C, in an attempt to cheat her.

Output: C verifies the proof. A wins the game if P was a valid proof.

III. PROPOSED SCHEME

We first provide the basic RDPC scheme only for static data integrity checking. Then, we show the advanced RDPC scheme that supports fully dynamic block operations based on TRT.

A. BASIC CAPDP SCHEME

The proposed scheme is based on an RSA cryptographic accumulator [10] that satisfies the characteristics and security requirements mentioned in Section II. It is composed of two phases: a setup phase and a proof generation-verification phase (PGV).

1) SETUP PHASE

This phase is a one-time operation executed by the data owner C before outsourcing the data to the CSP. C generates two symmetric keys k_H and k_E known only to C. Next, C splits the data M into n encrypted segments with ℓ_1-bits each using k_E, i.e., $M = \{m_1, \cdots m_n\}$, where m_i is the integer value of the encrypted i^{th} segment. A block generation (BlockGen) algorithm is used to generate an ℓ_2-bit tag τ_i for each segment m_i such that $\tau_i = H(m_i||k_H)$, where i is the segment index. Then, each tag is appended to its corresponding segment m_i to generate a precomputed data block b_i such that $B = \{b_1, \cdots, b_n\}$. C saves a copy (τ_i^s) of each tag (τ_i) in the TRT for use in subsequent verifications. Finally, C computes the accumulation value for set B using (2), i.e., $acc_B = g^{\prod_{i=1}^{n} b_i} \bmod N$. To compute acc_B, the product of the exponent is transformed into a series of modular exponentiations such that: $acc_B = (((g^{b_1})^{b_2})^{b_3})^{\cdots} \bmod N$ (i.e., single modular exponentiation) is executed for each element in set B.However, as the size of B increases or for large operands, this computation quickly becomes infeasible. For a one u-bit number and a v-bit exponent, the cost of modular exponentiation is $O(vM(u))$, where $M(u)$ is the time required to multiply two u-bit integers [31]. Consequently, computing acc_B would cost $O(n\ell M(\ell))$, where $\ell = \ell_1 + \ell_2$. According to Euler's totient theorem [31], the exponent of g in acc_B is equivalent to $\prod_{i=1}^{n} b_i \bmod \phi(N)$, where $\phi(N) = (p-1)(q-1)$. Given $\phi(N)$, the exponent of g could be easily computed by reducing it to mod$\phi(N)$ after each multiplication process, then by calculating single modular exponentiation in the end, i.e., $acc_B = g^{\prod_{i=1}^{n} b_i \bmod \phi(N)} \bmod N$. The computation cost of acc_B would, therefore, become $O(nM(\ell))$. Using this same argument, the witness w_j could also be computed easily as will be explained later in Section III-A-2. Hence, the knowledge of $\phi(N)$ allows the data owner and CSP to easily compute acc_B and w_j respectively. However, the data owner does not trust the CSP with the knowledge of $\phi(N)$ that can allow a dishonest CSP to pass a challenge request without holding the data. The following scenario illustrates how this could happen:

1) To outsource the data to the CSP, the data owner would first compute a prime representative $r(x)$ for each block $b_i \in B$ to make it collision-free, as discussed in Section II-A, such that $b'_i = r(b_i)$ and $B' = \{b'_1, \cdots, b'_n\}$.
2) The data owner then computes $acc_{B'} = g^{\prod_{i=1}^{n} b'_i \bmod \phi(N)} \bmod N$ and outsources B' to the CSP.
3) Once the CSP receives B', it computes the same accumulation value $acc_{B'}$.
4) When the data owner challenges the CSP with a random block index j, the CSP could use $\phi(N)$ and ${b'_j}^{-1}$, which is the modular multiplicative inverse of $b'_j \bmod \phi(N)$ (and can be found using the extended Euclidean algorithm [24]), to compute the witness w_j without the need to accumulate all elements in B' except b'_j, as shown below:

$$w_j = (acc_{B'})^{{b'_j}^{-1}} \bmod N = g^{{b'_j}^{-1} \prod_{i=1}^{n} b'_i \bmod \phi(N)} \bmod N \tag{4}$$

From this scenario, we can infer the main drawbacks of using prime representatives to represent the precomputed data blocks $B = \{b_1, \cdots, b_n\}$, summarized in the following two points:

1) Despite using collision-free prime representatives as the actual inputs to the accumulator, the computation of the accumulation value becomes increasingly infeasible as the number of inputs increases. This is due to the large number of modular exponentiations involved in the computation process. To avoid this problem, the CSP could be allowed to use $\phi(N)$ to compute the accumulation value. Unfortunately, this would also

Setup:

1) Two symmetric keys k_H and k_E are generated, which are known only to C.
2) C splits M into ℓ_1-bits encrypted segments by using k_E ; $M = \{m_1, \cdots m_n\}$.

BlockGen:

3) C generates ℓ_2-bits tag τ_i for each segment such that $\tau_i = H(m_i \| k_H)$, where i is the segment index.
4) C saves a copy (τ_i^s) of each tag (τ_i) in the TRT.
5) Each tag τ_i is appended to its corresponding segment (m_i) to generate a precomputed data block $b_i = m_i \| \tau_i$.
6) Each block b_i is left shifted by one bit, to generate the non-coprime representative (the data block) $b_i'' = b_i << 1$
7) C computes $acc_{B''} = g^{\prod_{i=1}^{n} b_i'' \bmod \phi(N)} \bmod N$ and stores it on her side. Then outsources B'' and $\phi(N)$ to the CSP

FIGURE 1. Setup phase.

allow an unreliable CSP to compute the accumulation value without actually holding the data.

2) A prime representative $r(x)$ for a value x is generally generated by computing the digest $H(x)$ as described by Bari? and Pfitzmann [32], where H is a one-way hash function. This means that the data owner cannot retrieve the data block b_i from its prime representative b_i'.

To overcome both drawbacks, we introduce the notation for the non-coprime representative of the precomputed data blocks $NCR(b_i)$. A non-coprime representative maps each b_i to a non-coprime number b_i'' using $\phi(N)$. According to [33], the multiplicative inverse of an integer $a \bmod N$ exists if and only if a and N are coprime ($\gcd(a, N) = 1$). This means that a non-coprime representative b_i'' of a precomputed data block b_i does not have a modular inverse with $\phi(N)$, which would force the CSP to use all the data blocks except block j to compute w_j. However, using non-coprime numbers instead of prime numbers as inputs to the accumulator does not meet the ''collision-free'' requirement and would allow the CSP to factor b_i'' into two factors and exponentiate one onto the witness w_j, providing the second as a response to the verification request, which allows the verification process to work successfully. To prevent such an occurrence, we propose the BlockGen algorithm, which is executed by the data owner to both generate the non-coprime representatives B'' and satisfy the ''collision-free'' requirement. As already mentioned, every encrypted segment (m_i) is appended to its corresponding tag τ_i to generate a precomputed data block $b_i = m_i || \tau_i \ \forall b_i \in B$. Using the BlockGen algorithm, each block b_i would instead be left-shifted by one (multiplied by 2) to make it non-coprime with $\phi(N)$, i.e. $b_i'' = b_i \ll 1$. This is true because $\phi(N)$ is an even number when $N \geq 3$ [31] and because the greatest common divisor between any two even numbers is also an even number. Hence, b_i'' and $\phi(N)$ are not relatively prime. When b_i is left-shifted by one (multiplied by 2), it's value could exceed the value of the RSA modulus (N). Therefore, p and q are chosen such that $2 * b_i < N$. Note that each b_i'' (which is the final data block) is an ℓ-bit integer, where $\ell = \ell_1 + \ell_2 + 1$. This secure structure of b_i'' prevents any replace attacks from the CSP, as will be described later in Section III-A-2. Finally, the data owner would compute the accumulation value $acc_{B''}$ for the set $B'' = \{b_1'', \cdots, b_n''\}$ using (5) and store it on her side; then, she outsources B'' and $\phi(N)$ to the CSP. Fig. 1 demonstrates the setup phase.

$$acc_{B''} = g^{\prod_{i=1}^{n} b_i'' \bmod \phi(N)} \bmod N \tag{5}$$

2) PROOF GENERATION-VERIFICATION PHASE (PGV)

In this phase, the data owner challenges the CSP to prove the integrity of the outsourced data in its possession, as shown in Fig. 2. The CSP responds by generating a proof that it holds the complete and uncorrupted dataset using the ProofGen algorithm. This generated proof is then verified by the data owner using the ProofVer algorithm. Checking the integrity of the outsourced data would proceed as follows:

1) The data owner C challenges the CSP with a random block index j.
2) The CSP uses $\phi(N)$ to compute the witness w_j for the challenged block b_j'' by accumulating all the data blocks $B'' = \{b_1'', \cdots, b_n''\}$ except b_j'':

$$w_j = g^{\prod_{\substack{i=1 \\ i \neq j}}^{n} b_i'' \bmod \phi(N)} \bmod N \tag{6}$$

3) The CSP sends (w_j, b_j'') to C as proof of data possession.
4) Upon receiving (w_j, b_j'') from the CSP, C computes the accumulated value:

$$acc'_{B''} = w_j^{b_j''} \bmod N \tag{7}$$

5) C extracts b_j from b_j'' by right-shifting b_j'' by one bit: $b_j = b_j'' >> 1$. Recall that $b_j = m_j || \tau_j$ and $\tau_j = H(m_j || k_H)$.

ProofGen:

1) The data owner C challenges the CSP with a random block index j.
2) The CSP uses $\phi(N)$ to compute the witness w_j for the challenged block b_j'' using (6).
3) The CSP sends (w_j, b_j'') to C as proof of data possession.

ProofVer:

1) Upon receiving (w_j, b_j''), C computes $acc'_{B''} = w_j^{b_j''} \bmod N$
2) C extracts b_j from b_j'' by right shifting b_j'' by one bit: $b_j = b_j'' >> 1$. Recall that $b_j = m_j \,||\, \tau_j$ and $\tau_j = H(m_j \,||\, k_H)$.
3) C extracts the received encrypted block m_j' from the extracted b_j and uses it along with k_H to compute $\tau_j' = H(m_j' \,||\, k_H)$.
4) C uses the TRT to locate the index (h) of the received tag τ_j.
5) The proof holds if $acc'_{B''} = acc_{B''}$, and $\tau_j' = \tau_j = \tau_h^s$, i.e. $h = j$.

FIGURE 2. PGV phase.

6) C extracts the received encrypted segment m_j from the extracted b_j and uses it along with k_H to compute $\tau_j' = H(m_j||k_H)$.
7) C uses the TRT to locate the index (h) of the received tag τ_j.
8) To be qualified as valid, the proof should meet the following three conditions:
   a) The computed accumulated value $acc'_{B''}$ must match the stored accumulated value $acc_{B''}$, which ensures that the CSP must use all the data blocks to compute the accumulated value using (6). Note that the non-coprime condition on any $b_i'' \in B''$ prevents the CSP from finding the multiplicative inverse of $b_i'' \bmod \phi(N)$, which would allow it to compute the witness w_j without the need to accumulate all elements in B'' using (8).

$$w_j = (acc_{B''})^{b_j''^{-1}} \bmod N = g^{b_j''^{-1} \prod_{i=1}^{n} b_i'' \bmod \phi(N)} \bmod N \quad (8)$$

   b) The computed tag τ_j' must match the received tag τ_j which ensures that it is computationally infeasible for the CSP to find a witness (w) for any value ($a \notin B''$) such that $acc_{B''} = w^a \bmod N$. This means that the CSP cannot replace the data block (b_j'') with any other nonvalid block ($a \notin B''$) to compute the witness because b_j'' depends on the hash value τ_j (which is computationally infeasible to generate without k_H). Recall that k_H is a symmetric key known only to C.
   c) The received tag τ_j must match the stored tag τ_h^s, which implies that $h = j$. This condition ensures that the CSP sends the requested block b_j'' in step 2 without replacing it with any other valid block b_i''.

B. DATA DYNAMICS SUPPORT

In the previous section, we assumed that M is static data. However, in real-world scenarios, outsourced data are usually dynamic. Therefore, we supported dynamic data operations in our proposed scheme i.e. insert, delete and update whether for a single block or a batch of blocks. In the following discussion, we will assume that dynamic data operations are performed on a batch of blocks.

1) INSERT OPERATION

If the data owner C wants to insert a set of new contiguous or non-contiguous segments $\{m_h, \cdots, m_{h+z}\}$ where z is the number of the new segments, she would perform the following steps:

- C computes the ℓ_2-bit tags $\{\tau_h, \cdots, \tau_{h+z}\}$ that is correspondent to the new encrypted data segments $\{m_h, \cdots, m_{h+z}\}$ such that $\tau_i = H(m_i||k_H)$, $h \le i \le h+z$.
- C generates the data blocks $\{b_h'', \cdots, b_{h+z}''\}$as described in Section III-A-1.
- An insert request, consisting of the data blocks and its indexes $\{(b_h'', h), \cdots, (b_{h+z}'', h+z)\}$ is sent by C to the CSP.
- The CSP inserts the blocks in the requested positions and sends an insertion confirmation message to C.
- When C receives the insertion confirmation message, she adds a new row for each new data block $\{b_h'', \cdots, b_{h+z}''\}$ in its appropriate position in the TRT. The values of the BI and TV fields of the inserted rows are set to $\{h, \cdots, h+z\}$ and $\{\tau_h, \cdots, \tau_{h+z}\}$ respectively.

BI	TV
1	τ_1^s
2	τ_2^s
3	τ_3^s
4	τ_4^s
...	...
n	τ_n^s

a)

BI	TV
1	τ_1^s
2	τ_2^s
3	τ_{n+1}^s
4	τ_3^s
...	...
n	τ_{n-1}^s
n+1	τ_n^s

b)

BI	TV
1	τ_1^s
2	τ_2^s
3	τ_4^s
4	τ_5^s
...	...
n-1	τ_n^s

c)

BI	TV
1	τ_1^s
2	τ_2^s
3	$\tau_{3'}^s$
4	τ_4^s
...	...
n	τ_n^s

d)

FIGURE 3. Dynamic operations using the TRT. a) Initial state. b) Insert a tag at position 3. c) Delete a tag at position 3. d) Update a tag at position 3.

- Since the new blocks are added to the set B'', C updates the accumulation value $acc_{B''}$ to include the newly added blocks, i.e. $acc_{B''} = acc_{B''}^{\prod_{i=h}^{h+z} b_i'' \bmod \phi(N)} \bmod N$ and then deletes these blocks from its local storage.

2) DELETE OPERATION

If the data owner C wants to delete a data block b_h'', where h is the block's index, this block should be decumulated from the accumulation value $acc_{B''}$. However, the deaccumulation of b_h'' from $acc_{B''}$ is infeasible because of the non-coprime condition, i.e. neither the data owner nor the CSP can compute the multiplicative inverse of any data block $b_h''^{-1}$. To allow both of the CSP and the data owner to delete data blocks without the need for de-accumulating values, an accumulator-based register ($Accr$) is introduced. The $Accr$ is initialized by 1 and is used by the CSP to accumulate all the deleted blocks. To delete a set of contiguous or non-contiguous data blocks $\{b_h'', \cdots, b_{h+z}''\}$ where z is the number of the deleted blocks, the following steps should be performed:

- C locates the indexes $\{h, \cdots, h+z\}$ of the data blocks $\{b_h'', \cdots, b_{h+z}''\}$ in the TRT and sends a delete request consisting of the data blocks indexes to the CSP.
- The CSP uses the received indexes $\{h, \cdots, h+z\}$ to locate the blocks that need to be deleted $\{b_h'', \cdots, b_{h+z}''\}$.
- Before deleting the data blocks $\{b_h'', \cdots, b_{h+z}''\}$ from the set B'', the CSP updates the $Accr$ as follows: $Accr = Accr. \prod_{i=h}^{h+z} b_i'' \bmod \phi(N)$
- The CSP deletes $\{b_h'', \cdots, b_{h+z}''\}$ from B'' and sends a deletion confirmation message to C.
- Upon receiving the confirmation message, C deletes $\{\tau_h, \cdots, \tau_{h+z}\}$ from the TRT. Note that C cannot decumulate $\{b_h'', \cdots, b_{h+z}''\}$ from $acc_{B''}$.

3) UPDATE OPERATION

The update operation is a two-step operation; the deletion of the existing version of data blocks, and the addition of the newer version. If the data owner C wants to update a set of contiguous or non-contiguous blocks $\{b_h'', \cdots, b_{h+z}''\}$ where z is the number of the modified blocks, she should perform the following steps:

- To update the data blocks to the new version $\{b_h'', \cdots, b_{h+z}''\}$, C sends an update request consisting of the updated data blocks and their indexes $\{(b_h'', h), \cdots, (b_{h+z}'', h+z)\}$ to the CSP.
- The CSP uses the received indexes $\{h, \cdots, h+z\}$, to locate the positions of the data blocks that need to be updated $\{b_h'', \cdots, b_{h+z}''\}$.
- Before updating the set B'' with the new version of data blocks, the CSP updates the $Accr$ to include the old version of blocks as follows:

$$Accr = Accr. \prod_{i=h}^{h+z} b_i'' \bmod \phi(N).$$

- The CSP replaces the old version of data blocks with the new version in the requested positions and sends an update confirmation message to C.
- C updates the corresponding tags $\{\tau_h, \cdots, \tau_{h+z}\}$ of the updated blocks $\{b_h'', \cdots, b_{h+z}''\}$ in the TRT.
- Since there are updated data blocks in the set B'', C updates the accumulation value $acc_{B''}$ to include them, i.e. $acc_{B''} = acc_{B''}^{\prod_{i=h}^{h+z} b_i'' \bmod \phi(N)} \bmod N$ and then deletes these blocks from the local storage. Note that the $acc_{B''}$ still includes the old version of data blocks. Fig. 3 shows an example of data dynamics using the TRT.

To support the delete and update operations, a slight modification to the PGV phase is required. For any succeeding data integrity verifications that use a random block index j, the CSP would compute the witness w_j for the block b_j'' by accumulating all data blocks in B'' except b_j'' using (3), then raise w_j to the power of $Accr$, i.e. $w_j = w_j^{Accr} \bmod N$. Finally, the CSP sends (w_j, b_j'') to C as proof of data possession. It is clear that $w_j^{b_j''} \bmod N$ should match the accumulated value $acc_{B''}$ stored at the data owner side.

IV. DISCUSSION

In this section, we elaborate on how our proposed scheme satisfies the characteristics of the cloud data integrity schemes discussed in Section II-B.

A. BLOCK-LESS VERIFICATION (C1)

In our proposed scheme, the data owner does not need to retrieve all the data blocks from the CSP to perform the verification process. Instead, the data owner challenges the CSP with a random block index j to verify the integrity of M. The CSP computes the witness w_j and sends it along with block b''_j to the data owner for verification as described in Section III-A-2.

B. UNRESTRICTED CHALLENGE FREQUENCY (C2)

The proposed scheme allows the data owner to challenge the CSP an unlimited number of times. In our scheme, it would be pointless for a CSP to cache the witness of the challenged block and use it to respond to a new challenge matching it, even though it could. The CSP might cache block witnesses for any of the following three reasons: 1) to hide an incident loss or any corruption of the outsourced data; 2) to spare more storage space, or 3) to reduce the computational cost of computing the witness. Below, we describe how our scheme addresses each case:

- **Hiding an incident loss or any corruption of the outsourced data:** Caching the witnesses would be neither an optimal nor a practical method for hiding any data loss or corruption. In our scheme, because the CSP must send the witness along with the requested block as a proof for data integrity, if the requested block is corrupted or deleted, the data owner can easily recognize that fact, as mentioned in Section III-A-2. Even if the corrupted block was not the requested block, the CSP caches the witnesses of all the blocks to hide any data corruption and deceives the data owner. Since the data blocks and their corresponding witnesses have the same size, caching the witnesses along with the uncorrupted data blocks would increase the storage overhead on the CSP. Instead, the CSP would store a backup copy of the outsourced data using the same storage space that it would have used if it had cached the witness of each data block.
- **Increasing available storage space by deleting all the outsourced data:** In our scheme, because the CSP must send the witness along with the requested block as a proof for data integrity, the CSP cannot delete the data blocks.
- **Reducing the computational cost of computing the witness:** Although caching the witnesses reduces the computational cost for the CSP, its storage cost would increase due to the extra storage space needed to save both the data blocks and their corresponding witnesses. Hence, CSP would not benefit by reducing the computational cost at the expense of the storage cost because the main objective of CSP is to offer more storage space to its clients.

C. SOUNDNESS (C3)

As discussed in Section IV-B, the assumption that the CSP can store the witness of the challenged block and use it to respond to a new challenge matching it is not a realistic assumption. In addition, due to the non-coprime condition on any $b''_i \in B''$, an unreliable CSP could not pass a challenge request without actually holding the data. Therefore, it would be computationally infeasible for the CSP to compute the witness w_j without accumulating all the elements in B''. Additionally, the CSP could not pass a challenge request with corrupted data because the witness w_j of the challenged block b''_j would have been computed by accumulating all the blocks, including the corrupted blocks. Thus, the resulting computed accumulated value $acc'_{B''}$ would not match the accumulated value $acc_{B''}$ stored at the data owner's side.

D. STATELESS VERIFICATION (C4)

Each challenge request in our proposed scheme is independent of the previous challenges with respect to both the CSP and the data owner; a data owner can challenge the CSP with a randomly chosen block index regardless of any prior verifications. Likewise, on the CSP side, witness computation is based exclusively on the current randomly chosen index. Accordingly, the computed witness of the challenged block is independent of any previously computed witnesses.

E. ROBUSTNESS (C5)

In the proposed scheme, any change of even a single bit in the data invalidates the integrity check. Because the outsourced file is divided into n segments converted to integers, any change to any bit in the n segments would lead to a different integer. Consequently, when the CSP computes the witness, the witness will also be altered. The CSP knows that the data owner will compute the accumulated value $acc'_{B''}$ based on the received proof (w_j, b''_j) and compares it with the stored value $acc_{B''}$. Accordingly, the CSP can try to deceive the data owner by modifying the accumulated value to match the one stored at the data owner's side. Fortunately, however, this process is infeasible for CSP because the data owner extracts the encrypted segment m_j from the received block b''_j, computes its tag using the secret key k_H, $\tau'_j = H(m_j||k_H)$, and then compares it with the received tag τ_j extracted from the received block b''_j to cross-check the validity of the block. Finally, the data owner will locate the index (h) of the received tag τ_j in the TRT to ensure that the received block is the requested block. This proves the robustness property of the proposed scheme.

F. DATA RECOVERY (C6)

As discussed earlier in Section III-A-2, data corruptions could be easily detected in our scheme. To support data recovery along with error-detection, error-correcting code based schemes, e.g. [34], could be easily integrated into our scheme.

G. DYNAMIC DATA HANDLING (C7)

As was previously discussed in Section III-B, the proposed scheme can handle data dynamics. After any dynamic operation executed on a single block or multiple blocks, the proposed integrity scheme remains intact. In addition, the proposed scheme does not require the retrieval of all the data blocks to execute any of the dynamic operations on the outsourced data.

H. PUBLIC AUDITABILITY (C8)

The proposed scheme supports the delegation of the responsibility for integrity verification to a trusted third-party auditor without the need to retrieve the outsourced data. To perform this process, the following three values must be transferred to the TPA: $acc_{B''}$, TRT and the hash key k_H. We assume that the transmission channel between the TPA and the data owner is secured with the SSL protocol. Note that, in such cases, the key k_H would be a symmetric key generated per-TPA to prevent the TPA from having access to any confidential data that belongs to the data owner. To delegate the verification process to the TPA, the data owner C would perform the following steps:

1) C sends a signed request with her private key PR_C along with her public key certificate $Cert_C$ to the TPA:
$$R_1 : (E_{PR_C}(ID_C), Cert_C).$$
2) The TPA verifies C's signature and replies with
$$R_2 : (E_{PR_{TPA}}(ID_C, ID_{TPA}), Cert_{TPA}).$$
3) C sends a signed request with her private key (PR_C) to the CSP. The request includes C's ID, the TPA's ID, the public key certificate of the TPA ($Cert_{TPA}$), a timestamp and a verification permission flag (P) that could be either granted or denied:
$$R_3 : E_{PR_C}(ID_C, ID_{TPA}, Cert_{TPA}, TS_{TPA}, P).$$
4) Upon receiving the request, the CSP verifies C's signature and the TPA's certificate and then saves the received request as a record in its database.
5) To perform the data integrity check on behalf of the user, the TPA sends a data verification request including ID_C, ID_{TPA} and $Cert_{TPA}$ to the CSP.
$$R_4 : E_{PR_{TPA}}(ID_C, ID_{TPA}, Cert_{TPA}).$$
6) The CSP compares the received request against the stored record to either grant or deny the TPA's request.

I. PRIVACY-PRESERVING (C9)

When assigning responsibility for the verification process to the TPA, the data owner must ensure that the TPA cannot access any of her confidential data. Therefore, our proposed scheme encrypts the segments in M using the encryption key K_E, which would be known only to her.

J. FAIRNESS (C10)

The proposed scheme protects a reliable CSP against dishonest users who may accuse the CSP of tampering with the outsourced data [35], as follows:

1) The data owner (C) requests a certificate for the tag generation process ($Cert_{TG}$) from the certificate authority (CA).
2) The CA generates $Cert_{TG}$ and sends it to C.
3) C generates an ℓ_2-bit tag τ_i for each segment using the private key $PR_{TG}PR_{TG}$ instead of K_H such that $\tau_i = H(m_i||PR_{TG})$.
4) C saves a copy ($\tau_i^s \tau_i^s$) of each tag (τ_i) in the TRT.
5) Each tag τ_i is appended to its corresponding segment (m_i) to generate a precomputed data block $b_i = m_i||\tau_i b_i = m_i||\tau_i$.
6) Each block b_i is left-shifted by one bit to generate the non-coprime representative (the data block) $b_i'' = b_i \ll 1 b_i'' = b_i \ll 1$.
7) C computes $acc_{B''} = g^{\prod_{i=0}^{n} b_i'' \bmod \phi(N)} \bmod N$ and then outsources B'' and $\phi(N)$ to the CSP.
8) If a legitimate but dishonest C were to accuse the CSP of tampering with block b_j'', the law enforcement agencies (LEA) would require C to reveal the certified private key of the tag generation process $PR'_{TG}PR'_{TG}$ according to the key disclosure law [36].
9) To ensure the correctness of the revealed private key $PR'_{TG}PR'_{TG}$, the LEA challenges C with a random string (S) encrypted with the C's certified public key of the tag generation process PU_{TG} (i.e., $(E_{PU_{TG}}(S))$) and requires C to decrypt it using the corresponding private key $PR'_{TG}PR'_{TG}$.
10) The LEA requires the CSP to reveal the data block b_j''.
11) The LEA regenerates the block tag $\tau_j' = H(m_i||PR'_{TG})$ and then extracts the tag $\tau_j\tau_j$ from b_j'' and compares it with τ_j'.
12) Finally, $\tau_j\tau_j$ must match $\tau_j'\tau_j'$ to prove the credibility of the CSP.

V. SECURITY ANALYSIS

The proposed RDPC scheme is proven secure under the security model in Section II-E.

Theorem 1: A cannot forge a tag and send it to the data owner as if it were a valid tag.

Proof:

Setup: C generates two symmetric keys k_H and $k_E K_E$ known only to her, and the challenged file M is divided into n encrypted segments $M = \{m_1, \cdots m_n\}$

Query: A selects some encrypted file segments of its choice m_i where $1 \leq i \leq n$, and sends it to C. C performs the BlockGen algorithm to obtain a valid tag for each segment $\tau_i = H(m_i||k_H)$ and stores the tag in the TRT. Then, C concatenates the computed tags τ to its corresponding file segments $M = \{m_1, \cdots m_n\}$ as $b_i = m_i||\tau_i b_i = m_i||\tau_i$ and left-shifts them by one bit (i.e., $b_i'' = b_i \ll 1 b_i'' = b_i \ll 1$)

to generate the data blocks. Then, C sends the data blocks $B'' = \{b''_1, \cdots, b''_n\}$ to A.

Finally, C computes the accumulated value of all the blocks and stores it on her side $acc_{B''} = g^{\prod_{i=0}^{n} b''_i \bmod \phi(N)} \bmod N$.

Challenge: C challenges A with a random block index j and waits for the proof $P = (w_j, b''_j)$.

Forge: First, A computes the accumulator $acc_{B''}acc_{B''}$ for all the outsourced blocks and caches the value to deceive the challenger C in the future. Upon receiving the challenge, if all the blocks are deleted, A will forge the proof $P = (\bar{w}_j, \bar{b''}_j)$ in such a way as to ensure they match the cached $acc_{B''}acc_{B''}$ (i.e., $acc_{B''} = \bar{w}_j^{\bar{b''}_j} \bmod N$). However, for deleted segments and valid tags, A will forge block $\bar{b''}_j$ by concatenating a forge segment $\bar{m}_j\bar{m}_j$ to a valid tag $\tau_j\tau_j$, making both blocks $\bar{b''}_j$ and $\bar{w}_j$ match the cached $acc_{B''}acc_{B''}$. Then, A will send the forged proof P as a valid proof to C in an attempt to cheat her.

Output: When C receives the proof $P = (\bar{w}_j, \bar{b''}_j)$, she computes $acc'_{B''} = \bar{w}_j^{\bar{b''}_j} \bmod N$ and checks whether $acc'_{B''}acc'_{B''}$ matches $acc_{B''}acc_{B''}$. There is some probability that A can win the game at this step. Therefore, to ensure the data is intact, C extracts the encrypted segment $\bar{m}_j\bar{m}_j$ from $\bar{b''}_j$ and computes its tag $\tau'_j = H(\bar{m}_j||k_H)$. After that, C compares $\tau'_j\tau'_j$ with the received tag $\bar{\tau}_j\bar{\tau}_j$ which is extracted from $\bar{b''}_j$. Because k_Hk_H is secret, it is extremely unlikely that $\tau'_j = \bar{\tau}_j\tau'_j = \bar{\tau}_j$. Therefore, A cannot succeed in deceiving C.

Theorem 2: A cannot replace any damaged or deleted block with another valid block to deceive C.

Proof:

Challenge: C challenges A with a random block index j and waits for the proof $P = (w_j, b''_j)$.

Forge: A replaces the proof with a valid proof $P = (w_k, b''_k)$ for another block b''_k and sends it to C in an attempt to cheat her.

Output: When C receives the proof $P = (w_k, b''_k)$, she computes $acc'_{B''} = w_k^{b''_k} \bmod N$ and checks whether $acc'_{B''}acc'_{B''}$ matches $acc_{B''}acc_{B''}$. Then, C computes the tag $\tau'_k = H(m_k||k_H)$ and compares it with the received tag $\tau_k\tau_k$. Thus far, A will win the game. Therefore, to ensure the data is intact, C checks whether the computed tag $\tau'_k\tau'_k$ is the requested tag $\tau_j\tau_j$ by using the TRT to ensure that no replacements have been made. Hence, A cannot succeed in deceiving C.

Theorem 3: A cannot use an old challenge response to respond to a new challenge that matches it to deceive C.

Proof: Consider that A deleted all the outsourced data blocks and cached their witnesses ww and the $acc_{B''}acc_{B''}$.

Challenge: C challenges A with a random block index j and waits for the proof $P = (w_j, b''_j)$.

Forge: A retrieves the cached witness w_j for the requested block and forges the block $\bar{b''}_j$, making both $\bar{b''}_j$ and w_j match the cached $acc_{B''}acc_{B''}$. Then, A sends $P = (w_j, \bar{b''}_j)$ to C in an attempt to cheat her.

Output: When C receives the proof $P = (w_j, \bar{b''}_j)$, she computes $acc'_{B''} = w_j^{\bar{b''}_j} \bmod N$ and checks whether $acc'_{B''}acc'_{B''}$ matches $acc_{B''}acc_{B''}$. Then, C computes the tag $\tau'_j = H(\bar{m}_j||k_H)$ and compares it with the received tag $\bar{\tau}_j\bar{\tau}_j$. Because k_Hk_H is secret, it is highly unlikely that $\tau'_j = \bar{\tau}_j\tau'_j = \bar{\tau}_j$. Therefore, A cannot succeed in deceiving C.

But in case the CSP deletes some not all data blocks and the data owner challenged one of the undeleted blocks, the CSP could cache the witnesses of the undeleted blocks in order to deceive the data owner. In this case, the CSP would be doubling its storage cost since the block size and the witness size are the same. In order to gain storage benefits from the deletion of blocks, the CSP should delete more than 50% of the outsourced data. For example, if the CSP wants to save 10% of the storage space occupied by the outsourced data, it should delete 55% of the data blocks and cache the witnesses of the remaining 45% along with their corresponding data blocks (≈90% of the outsourced data). However, by deleting more data blocks, the CSP risks higher chances of being detected by the data owner challenging a deleted block. In order to check if the CSP is deceiving the data owner, we simulated replay attack against the proposed scheme to determine the number of challenges required for the data owner to detect any CSP falsification in the case of deletion of 10% to 95% of the outsourced data. As observed in Table 2, as the CSP deletes more blocks, the probability of the data owner in detecting the occurrence of a replay attack increases. In specific, if the CSP deletes more than 50% of the outsourced data, the data owner would be able to detect the occurrence of the attack either on the first challenge or the second challenge at most. But if the deleted blocks are less than 50%, then the probability of the data owner's early detection decreases. However, due to the two-fold increase in the storage space on the CSP side caused by caching the witnesses, this latter attempt is highly unlikely (or unfavorable for the CSP).

TABLE 2. Required challenges to detect replay attacks.

Percentage of deleted blocks (CSP)	Number of challenges (Data Owner)
10%	9
25%	3
40%	2
55%	2
70%	1
85%	1
95%	1

Theorem 4: A cannot reveal the outsourced data during the verification process (i.e., no data leakage will occur).

Proof: In our proposed scheme, all the data segments are encrypted using the encryption key k_EK_E, which is known only to the data owner; therefore, there is no possibility of data leakage.

As shown in the previous theorems, our proposed scheme prevents tag forgery, data deletion, replacement, data leakage attacks and detects replay attacks.

VI. PERFORMANCE ANALYSIS

This section analyses the performance of the proposed scheme regarding computation cost, storage cost and communication cost from the viewpoint of the CSP and the data owner. To evaluate the efficiency of the proposed scheme, a prototype of the proposed scheme is introduced. Then, it is followed by a comparison with other deterministic and probabilistic schemes which are Hao's scheme [19] which is applying the deterministic data integrity check, Yong Yu's scheme [11] which is applying the probabilistic check and Mingxu Yi's scheme [20] which is applying both types of data integrity check. Table 3 shows a list of notations representing operation costs used throughout this section.

TABLE 3. Notations of operation costs.

Symbol	Operation cost
T_{mod}	Modular operation
T_{mul}	Modular multiplication
T_{add}	Modular addition
T_{exp}	Modular exponentiation
T_{enc}	Symmetric encryption
T_{prng}	Pseudo-random number generator
T_{hash}	Hash function computation
T_{perm}	Pseudo-random permutation function
T_{pair}	Pairing operation

A. COMPUTATION COST

The proposed scheme is composed of two phases: the setup phase and the PGV phase. The setup phase is a one-time operation which is performed at the data owner's side. Through the setup phase, the file M is divided into n segments which are encrypted, hashed, and left shifted by one bit to generate the data blocks that will be accumulated as discussed in Section III-A-1. The computation cost in the setup phase can be expressed as $nT_{enc} + nT_{hash} + nT_{mul} + T_{\text{exp}}$. In the PGV phase, the data owner computes the accumulated value $acc'_{B''} = w_j^{b''_j} \bmod N$, where b''_j is the challenging block, so, the computation cost of the verification operation is T_{exp}. The CSP's computation cost is based on (6) whose computation cost is $(n-1)T_{mul} + T_{\text{exp}}$. For data dynamics, the computation cost is presented from the viewpoint of both the CSP and the data owner. In the insert and update operations, the computation cost at the data owner side is $dT_{enc} + dT_{hash} + dT_{mul} + T_{\text{exp}}$ where d is the number of the new inserted/updated blocks. There is no computation overhead on the CSP during the insert operation. Whereas in the update and delete operations, the computation overhead on the CSP is dT_{mul}, where d is the number of the deleted/updated blocks that the CSP needs to accumulate to update $Accr$ as mentioned in Sections III-B-2 and III-B-3. While at the data owner side, there is no any computation overhead during the delete operations. As the proof generation will be slightly modified as mentioned before, the computation cost of the proof generation operation will be $(n-1)T_{mul} + 2T_{\text{exp}}$. The additional modular exponentiation operation is added due to raising w_j to the power of $Accr$ as discussed in Section III-B-3.

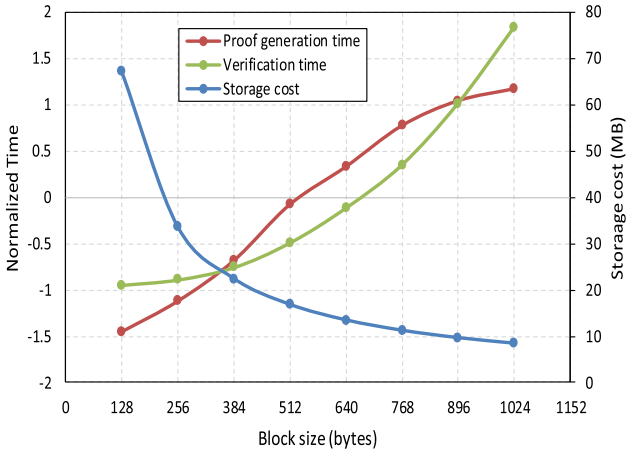

FIGURE 4. Trade-off between storage cost and both proof generation and verification time for 1 GB file.

B. STORAGE COST

In the proposed scheme, the data owner needs storage space to store the TRT, $acc_{B''}$, k_H and k_E. Regarding the CSP, the storage cost consists of the file, tags, and $Accr$. Therefore, the storage cost at the data owner's side is $|TRT| + |acc_{B''}| + |k_H| + |k_E|$ where at the CSP's the storage cost is $|M| + n|\tau| + |Accr|$. Note that $|TRT| = n|\tau_i|$.

C. COMMUNICATION COST

The communication cost in our proposed scheme consists of two parts: the challenge sent from the data owner to the CSP which is a random block index (j) and the response sent back from the CSP to the data owner $\left(w_j, b''_j\right)$. So, the communication cost is $|w_j| + \left|b''_j\right| + |j|$. Regarding data dynamic operations, the communication cost of the insert or update operations is $d\left|b''_j\right|$, where in the delete operation the communication cost is $d|j|$. Table 4 shows a performance comparison between the proposed scheme, Hao's, Yong Yu's, and Mingxu Yi's schemes.

D. PROTOTYPE IMPLEMENTATION

The prototype for the proposed scheme CAPDP is implemented using the C++ programming language with GMP library [37] and Crypto++ Library [38]. The AES-128 [39] algorithm is used for block encryption and SHA-3 algorithm with variable output length [40] for generating collision-free

TABLE 4. Performance comparison between the proposed scheme and other schemes.

Costs		Proposed Scheme	Hao's Scheme [19]	Yong Yu's Scheme [11]	Mingxu Yi's Scheme [20]	
					Probabilistic	Deterministic
Computation	Setup	$nT_{enc}+nT_{hash}+nT_{mul}+T_{\exp}$	$nT_{enc}+nT_{\exp}$	$nT_{mul}+nT_{hash}+2nT_{\exp}$	$m^cT_{enc}+2\tau^cT_{prng}+mn[(\tau+2)T_{mul}+(\tau+1)T_{\exp}+(\tau+1)T_{add}+T_{hash}]$	
	Challenge Gen.	T_{prng}	$T_{\exp}+2T_{prng}$	$c^b(T_{perm}+T_{prng})$	$mc(T_{perm}+T_{prng})$	mnT_{prng}
	Proof Gen.	$(n-1)T_{mul}+T_{\exp}$	$n(T_{prng}+T_{mul}+T_{\exp}+T_{\mathrm{mod}})+T_{\exp}$	$c(T_{perm}+T_{prng}+T_{add}+T_{\exp}+2T_{mul})$	$mc(T_{\exp}+T_{mul})$	$mn(T_{\exp}+T_{mul})$
	Verification	$T_{\exp}$	$nT_{prng}+(n+1)T_{\exp}+nT_{mul}$	$cT_{hash}+(2c+1)T_{\exp}+2cT_{mul}$	$2T_{pair}+(mc\tau+mc+\tau+1)T_{mul}+T_{\exp}+(m\tau+mc+mc\tau+mc\tau^2)T_{add}+mc\tau T_{hash}$	$2T_{pair}+(mn\tau+mn+\tau+1)T_{mul}+T_{\exp}+(m\tau+mn+mn\tau+mn\tau^2)T_{add}+mn\tau T_{hash}$
Communication		$\lvert j\rvert+2\lvert N\rvert$	$\lvert r\rvert^b+2\lvert N\rvert$	$\lvert c\rvert+12\lvert N\rvert+2\ell_h+\lvert r\rvert*\lvert N\rvert$	$mc(\ell^c)+mc\lvert r\rvert+\lvert c\rvert+\ell^c{}_2$	$mn(\ell)+mn\lvert r\rvert+\ell_2$
Storage	Data owner	$n\ell^a{}_h+\lvert N\rvert+\lvert K_H\rvert+\lvert K_E\rvert$	$n\lvert N\rvert$	—	$2\tau\lvert r\rvert$	
	CSP	$\lvert M\rvert+n\ell_h+\lvert N\rvert$	$\lvert M\rvert$	$\lvert M\rvert+n\lvert N\rvert+6\lvert N\rvert+\ell_h$	$m\lvert M\rvert+n\ell_2$	

[a] ℓ_h represents the message digest size.

[b] r represents a random number and c represents the number of challenging blocks.

[c] In Mingxu Yi's scheme, m represents the number of all file copies, τ represents the number of sectors in each block, ℓ represents the block size, and ℓ_2 represents the tag size.

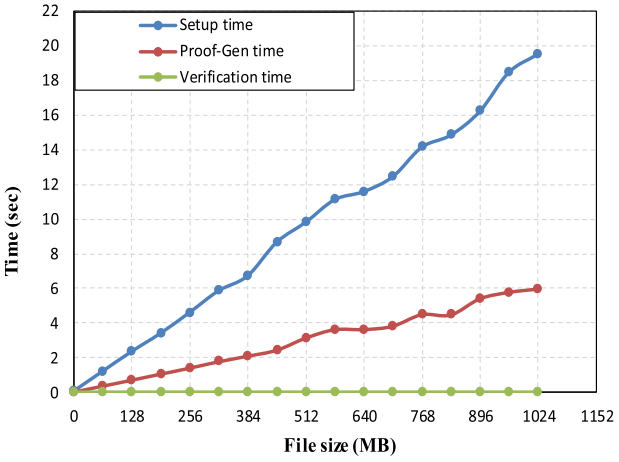

FIGURE 5. The computation cost of the proposed scheme computed on different file sizes and a 384-byte block size.

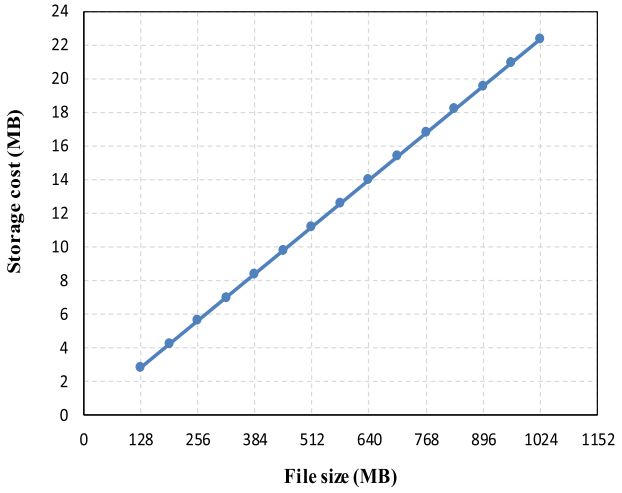

FIGURE 6. The storage cost of the proposed scheme computed on different file sizes and 384-byte block size.

64-bit tags. The implementation is executed on a Lenovo Laptop Z50 with Ubuntu operation system, Core i7-4510U @2.60 GHz CPU, 8G. The optimal block size used in Hao's, Yong Yu's, and Mingxu Yi's schemes are 8 KB, 4 KB, and 8 KB respectively. To figure out the optimal block size of the proposed scheme, we analysed the trade-off between the storage cost from one side and the proof generation time and ver-

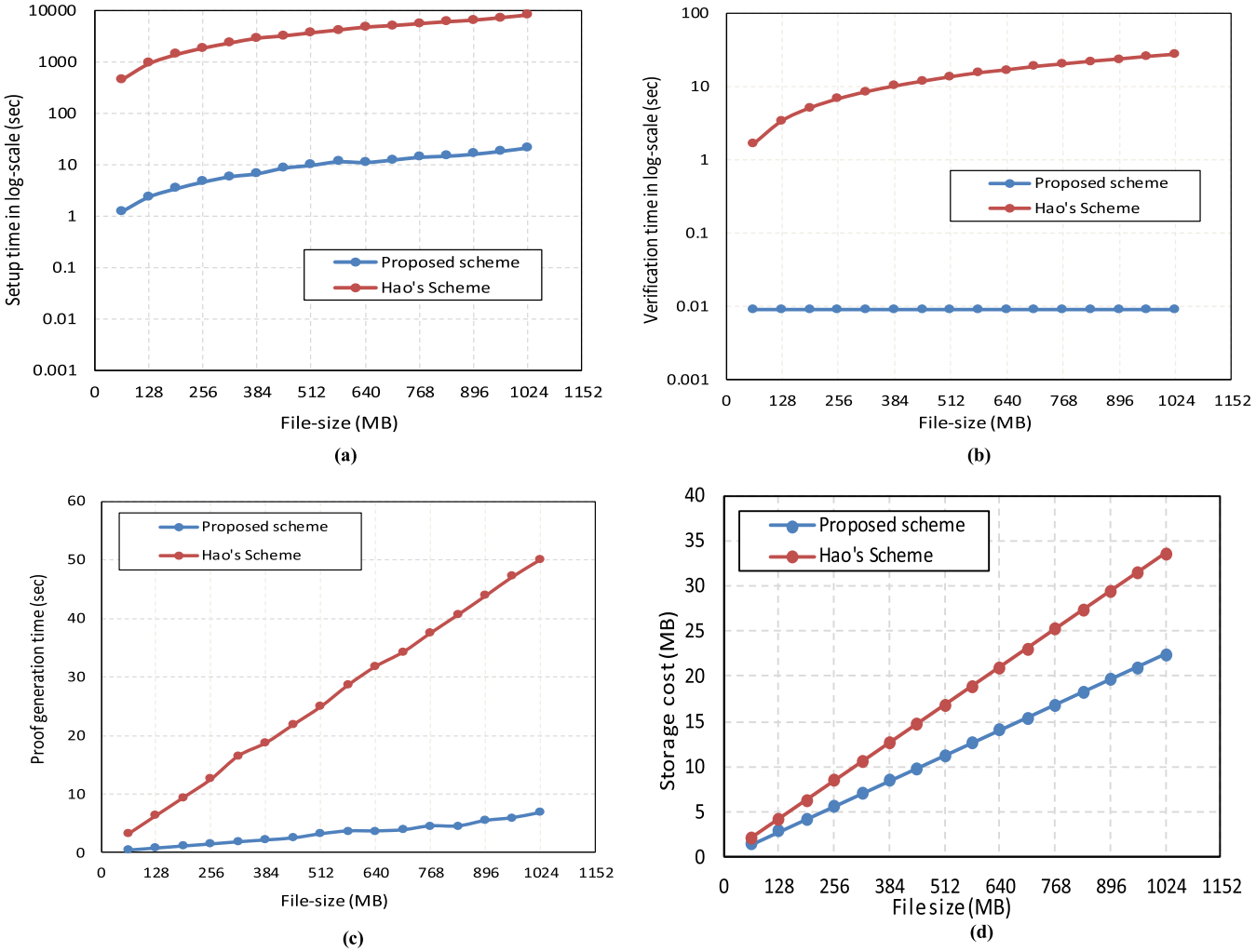

FIGURE 7. The computation and storage costs of the proposed and Hao's schemes on different size of files. (a) Setup time comparison. (b) Verification time comparison. (c) Proof generation time comparison. (d) Storage cost comparison.

ification time from the other side, for a fixed file size of 1 GB and different block sizes from 128 to 1024-byte, with an increment of 128-byte for each step. From Fig 4., we can see that the best trade-off between the storage cost and both the proof generation and verification time is achieved when the block size is closed to 384. Thus, we choose 384-byte as the optimal block size in all experiments. To provide a proper basis for comparison, the proof generation and verification time values are normalized using z-score normalization [41] as shown in Fig. 4. In the implementation phase, we measured the computation cost (i.e. setup time, proof generation time, and verification time), and the storage cost of the proposed scheme for all file sizes that ranged from 64 MB to 1 GB, with the increments of 64 MB. The results of these measurements are shown in Fig. 5 and Fig. 6. As we can see, the setup time, proof generation time and storage cost increase almost linearly with the increase of the file size whereas the verification time remains constant (9 ms) and independent of the file size. In this phase, we also evaluated and compared the computation and the storage costs of our scheme versus Hao's scheme for all file sizes that ranged from 64 MB to 1 GB, with the increments of 64 MB. The experimental results are shown in Fig. 7. Due to the massive difference between the two schemes, the setup and verification time are plotted in log scale, as shown in the figure. Additionally, a performance comparison between both schemes, in terms of, computation, communication, and storage costs for a 1 GB file, is shown in Table 5. As shown in Fig. 7-a, the setup time of Hao's scheme is much higher than that of the proposed scheme. For a 1 GB file, the setup time for the proposed scheme and Hao's scheme is 19 and 15205 s, respectively (≈99.9% reduction in the setup computation time), see Table 5. In Fig. 7-b, the verification time is constant (9 ms), whereas in Hao's scheme it increases as the file size increases. This is because the verification operation, performed by the data owner during the PGV phase, is independent of the file size, unlike the verification process in Hao' scheme which is dependent on the file size. Therefore, the verification operation of the

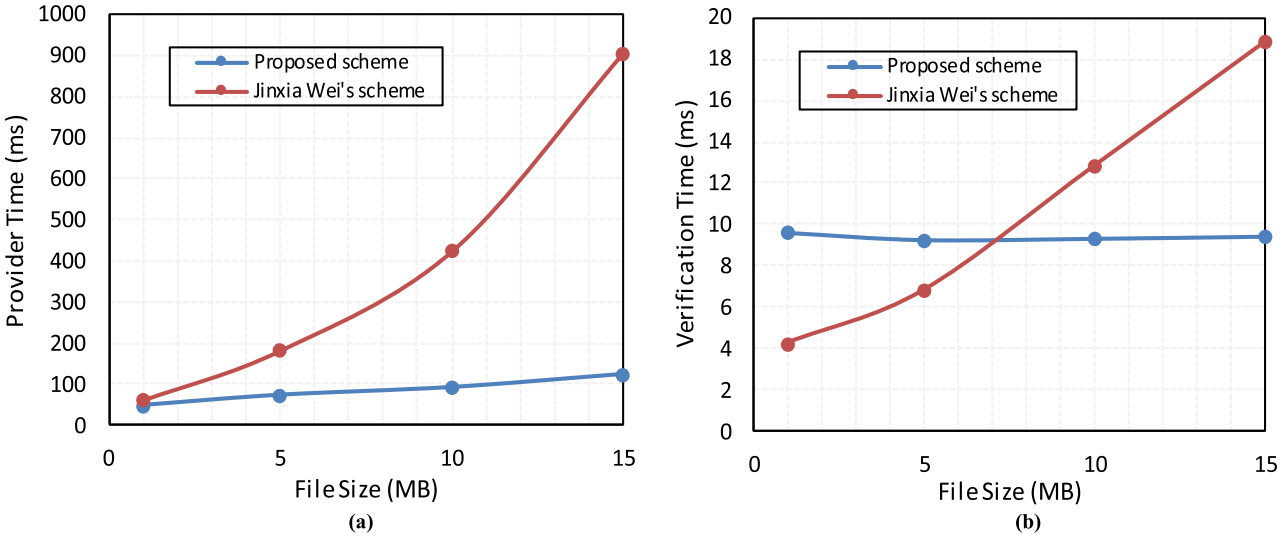

FIGURE 8. The computation costs of the proposed and Mingxu Yi's schemes on different size of files. (a) Proof generation time comparison. (b) Verification time comparison.

TABLE 5. Performance of the proposed scheme and Hao's scheme on 1 GB file.

Costs		Proposed Scheme	Hao's Scheme
Computation	Setup	19 *s*	15205 *s*
	Challenge Gen.	0.8 μs	4.8 μs
	Proof Gen.	6 *s*	50 *s*
	Verification	0.0095 *s*	57 *s*
Communication		776 *bytes*	784 *bytes*
Storage	Overhead	21.3 *MB*	48 *MB*
	Percentage	2.1%	4.7%

proposed scheme is lightweight and can be performed by a low-power data owner regardless of the file size. As observed in Fig. 7-c, the proof generation time of the proposed scheme is much lower than that of the Hao's scheme, where in the latter scheme the proof generation time increases as the file size increase at a rate higher than that of the proposed scheme. For a 1 GB file, the proof generation time for the proposed scheme and Hao's scheme is 6 and 50 s, respectively (≈88% reduction in the proof generation computation time), see Table 5. The storage cost comparison between the proposed scheme and Hao's scheme is shown in Fig. 7-d. It is obvious that the storage overhead at the data owner's side in the proposed scheme is less than that of Hao's scheme. This is due to the storage overhead of Hao's scheme that grows with increasing file sizes at a rate higher than that of the proposed scheme. For a 1 GB file, the storage cost for the proposed scheme and Hao's scheme is 21.3 MB and 48MB, respectively (≈55.6% reduction in the storage cost), see Table 5.

In Mingxu Yi's scheme, the authors implemented their probabilistic scheme on different file sizes ranging from 1 MB to 15 MB with an increment of 5 MB and fixed block size 8 KB. By comparing our deterministic results on these different file sizes with their probabilistic results, we found that the proof-generation and verification time of Mingxu Yi's scheme is much higher than that of the proposed scheme as shown in Fig. 8. Moreover, in their deterministic scheme, in order to verify the integrity of the file, the CSP should send all the file blocks of each file copy again to the verifier, which will lead to a high communication cost that will grow as the file size increase. Therefore, our proposed scheme is proved to be more efficient in terms of computation and communication costs.

As for Yong Yu's scheme, it costs the CSP 22.36 ms and the data owner 219.54 ms to generate and verify a proof respectively for a 40 MB file size when challenging 460 blocks i.e. 4.6 % of the file [11]. Whereas in our proposed scheme it costs the CSP 249 ms and the data owner 9.5 ms when challenging all the file blocks i.e. 100 % of the file size. As shown above, the proof generation phase in our scheme takes more time than in Yong Yu's scheme since in our scheme we challenge all the file blocks not just a set of random blocks. This extra computation overhead should not cause problems to the CSP due to its high-power capabilities. On the other hand, our proposed scheme is better in terms of the verification cost which is an important feature, due to the data owner's limited resources i.e. there will be no burden on the data owner side.

VII. CONCLUSION

Public cloud storage is a cloud service that allows individuals and organizations to store and manage data remotely without being restricted to the capacity of their local storage devices. Public cloud storage is widely used by many crucial sectors,

including many of the top governmental agencies. These sectors require a 100% data integrity and possession guarantee of their outsourced data. In this paper, we proposed a deterministic private verification data integrity check scheme that efficiently provides such integrity and possession guarantees. The proposed scheme employs a modified RSA-based cryptographic accumulator to verify the integrity of the outsourced data, which limits the computational and storage overhead for both the CSP and the data owner to acceptable levels. Using our scheme, the data owner must store only the accumulator value (for verification purposes) and the TRT. To provide the minimal storage required at the data owner's side, fog nodes [42] can be employed such that the data owner uploads the TRT to a fog server which is in closer proximity to the data owners. Due to the added TRT feature, our scheme also supports dynamic data operations (insert, update and/or delete data). However, the scheme's main advantage is that it minimizes the burden and cost of the verification process on the data owner's side, enabling verification to be performed even on low-power devices and that is because the verification operation in our scheme is independent of the number of blocks being verified. In terms of security and performance, our scheme was able to prevent tag forgery, data deletion, replacement, data leakage attacks and detect replay attacks and shown to be efficient, practical and feasible in real-life applications. As future work, we think it is worth exploring further optimizations in our implementation to not just detect but prevent replay attacks completely.

REFERENCES

[1] Y. Zhang, R. Deng, X. Liu, and D. Zheng, "Outsourcing service fair payment based on blockchain and its applications in cloud computing," *IEEE Trans. Serv. Comput.*, to be published.
[2] F. Zafar *et al.*, "A survey of cloud computing data integrity schemes: Design challenges, taxonomy and future trends," *Comput. Secur.*, vol. 65, pp. 29–49, Mar. 2017.
[3] L. Taylor, "Which agencies are leading the way into the cloud?" *IEEE Cloud Comput.*, vol. 1, no. 4, pp. 78–82, Nov. 2014.
[4] M. Dekker, "Critical cloud computing-A CIIP perspective on cloud computing services," Eur. Netw. Inf. Secur. Agency (ENISA), Heraklion, Greece, White Paper, 2012.
[5] J. Miller. (2017). *Critical Data is More Secure in Cloud-Based Data Centers*. [Online]. Available: http://www.avidsolutionsinc.com/why-critical-data-is-more-secure-in-the-cloud
[6] (2017). *Amazon Web Services (AWS)*. [Online]. Available: https://aws.amazon.com/
[7] (2017). *NetApp*. [Online]. Available: http://www.netapp.com/us/index.aspx
[8] NetApp. (2012). *Agencies Built on NetApp Go Further, Faster*. [Online]. Available: http://www.netapp.com/us/media/agencies-built-on-netapp-go-further.pdf
[9] P. S. Kumar, "New probabilistic efficient and secure protocols for data storage security in cloud computing," Ph.D. dissertation, Dept. Comput. Sci., Pondicherry Univ., Puducherry, India, 2014
[10] E. Tremel, "Real-world performance of cryptographic accumulators," M.S. thesis, Dept. Comput. Sci., Brown Univ., Providence, RI, USA, 2013.
[11] Y. Yu *et al.*, "Cloud data integrity checking with an identity-based auditing mechanism from RSA," *Future Gener. Comput. Syst.*, vol. 62, pp. 85–91, Sep. 2016.
[12] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 78–88, Jan. 2017.
[13] R. Saxena and S. Dey, "Cloud audit: A data integrity verification approach for cloud computing," *Procedia Comput. Sci.*, vol. 89, pp. 142–151, 2016.
[14] G. Caronni and M. Waldvogel, "Establishing trust in distributed storage providers," in *Proc. 3rd Int. Conf. Peer-to-Peer Comput. (P2P2003).*, Sep. 2003, pp. 128–133.
[15] Y. Deswarte, J.-J. Quisquater, and A. Saïdane, "Remote integrity checking," in *Integrity and Internal Control in Information Systems VI*. Berlin, Germany: Springer, 2004, pp. 1–11.
[16] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," *IACR Cryptol. ePrint Arch.*, vol. 2006, 2006.
[17] F. Sebe, J. Domingo-Ferrer, A. Martinez-balleste, Y. Deswarte, and J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1034–1038, Aug. 2008.
[18] A. F. Barsoum and M. A. Hasan, "Provable possession and replication of data over cloud servers," Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep. 2010/32, 2010.
[19] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 9, pp. 1432–1437, Sep. 2011.
[20] M. Yi, J. Wei, and L. Song, "Efficient integrity verification of replicated data in cloud computing system," *Comput. Secur.*, vol. 65, pp. 202–212, Mar. 2017.
[21] X. Yi, M. Kaosar, M. Golam, R. Paulet, and E. Bertino, "Single-database private information retrieval from fully homomorphic encryption," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 5, pp. 1125–1134, May 2013.
[22] J. Benaloh and M. de Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Proc. Adv. Cryptol.-EUROCRYPT Workshop Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 1994, pp. 274–285.
[23] D. Derler, C. Hanser, and D. Slamanig, "Revisiting cryptographic accumulators, additional properties and relations to other primitives," in *Proc. Topics Cryptol.-CT-RSA Cryptographers Track RSA Conf.*, San Francisco, CA, USA: Springer, Mar. 2015, pp. 127–144.
[24] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.
[25] C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Authenticated hash tables based on cryptographic accumulators," *Algorithmica*, vol. 74, pp. 664–712, Feb. 2016.
[26] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Proc. Annu. Cryptol.-CRYPTO Annu. Int. Cryptol. Conf.* Santa Barbara, California, USA: Springer, Aug. 2002, pp. 61–76.
[27] M. T. Goodrich, R. Tamassia, and J. Hasić, "An efficient dynamic and distributed cryptographic accumulator," in *Proc. Inf. Secur. Int. Conf. Inf. Secur. (ISC)*. Sao Paulo, Brazil: Springer, Sep./Oct. 2002, pp. 372–388.
[28] T. Sander, "Efficient accumulators without trapdoor extended abstract," in *Proc. Inf. Commun. Secur. Int. Conf. Inf. Commun. Secur. (ICICS)*. Sydney, NSW, Australia: Springer, Nov. 1999, pp. 252–262.
[29] A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 485–497, Mar. 2015.
[30] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Alexandria, VA, USA, 2007, pp. 598–609.
[31] R. P. Brent and P. Zimmermann, *Modern Computer Arithmetic*, vol. 18. Cambridge, U.K.: Cambridge Univ. Press, 2010.
[32] N. Barić and B. Pfitzmann, "Collision-free accumulators and fail-stop signature schemes without trees," in *Proc. Adv. Cryptol.-EUROCRYPT Int. Conf. Theory Appl. Cryptograph. Techn.* Konstanz, Germany: Springer, May 1997, pp. 480–494.
[33] V. Shoup, *A Computational Introduction to Number Theory and Algebra*. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[34] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 43–54.
[35] Y. Zhang, R. H. Deng, X. Liu, and D. Zheng, "Blockchain based efficient and robust fair payment for outsourcing services in cloud computing," *Inf. Sci.*, vol. 462, pp. 262–277, Jun. 2018.
[36] T. Schreider, *The Manager's Guide to Cybersecurity Law: Essentials for Today's Business*. Brookfield, CT, USA: Rothstein, 2017.
[37] T. Granlund, *Gnu MP 6.0 Multiple Precision Arithmetic Library*. London, U.K.: Samurai Media Limited, 2015.
[38] W. Dai, *Crypto++ Library 5.6.0*. Crypto++ Community, 2009. [Online]. Available: https://www.cryptopp.com/

[39] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*. Berlin, Germany: Springer, 2013.
[40] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," Federal Inf. Process. Stds., Gaithersburg, MD, USA, Tech. Rep. (NIST FIPS)-202, 2015.
[41] R. J. Larsen and M. L. Marx, *An Introduction to Mathematical Statistics and Its Applications*, vol. 2. Englewood Cliffs, NJ, USA: Prentice-Hall, 1986.
[42] M. R. Anawar, S. Wang, M. A. Zia, A. K. Jadoon, U. Akram, and S. Raza, "Fog computing: An overview of big IoT data analytics," *Wireless Commun. Mobile Comput.*, vol. 2018, May 2018, Art. no. 7157192.

WALID I. KHEDR received the Ph.D. degree in computer science from Ain Shams University, in 2009. He is currently an Associate Professor in information technology with the Faculty of Computers and Informatics, Zagazig University. His current research interests include network security protocols, key management protocols, cloud security, and the Internet of Things security.

HEBA M. KHATER received the B.Sc. degree from the Faculty of Computers and Informatics, Department of Information Technology, Zagazig University, Egypt. Her interests include cloud computing, cryptography, computer networking, and network security.

EHAB R. MOHAMED received the B.Sc., M.Sc., and Ph.D. degrees in communication from the Faculty of Engineering, Zagazig University, where he is currently an Associate Professor with the Information Technology Department, Faculty of Computers and Informatics. His current research interests include optimization, computational intelligence, computer networks, image processing, and cloud computing.